Architecting OTA Update Systems for IoT: A Quality-Attribute–Driven Systematization

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Mónica Michelle Villegas Arias

Thesis Supervisor: Prof. Dr. Mauricio Solar

Thesis Co-Supervisor: Prof. Dr. Hernán Astudillo
Prof. Dr. Fáber D. Giraldo

Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Doctor en Ingeniería Informática

March 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☐ Memoria o trabajo de título; ☒ Tesis de Postgrado;

Título del trabajo: Architecting OTA Update Systems for IoT: A Quality-Attribute-Driven Systematization

Nombre del candidato(a): Mónica Michelle Villegas Arias

Carrera / Grado: Doctorado en Ingeniería Informática

Campus: Casa Central Valparaiso ; **Departamento**: Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, **Mauricio Solar** , en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: ; **Firma:** _____

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA · DEPARTAMENTO INFORMATICA

Mauricio Solar Fuentes

Estudiante o Candidato(a):

Fecha: 15/01/2026 ; **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por
estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

I would like to dedicate this thesis to my loving parents and sister, for their support, encouragement, and belief in me throughout this process. To my partner, whose love, patience, and support made even the most challenging moments manageable. I also extend my gratitude to my teachers for their guidance, knowledge, and dedication, which have shaped my academic and professional development. Finally, to my beloved cat, whose quiet companionship and unconditional affection provided comfort and balance during this journey.

Declaration

I declare that, unless otherwise referenced, this dissertation is original. Its contents have not been submitted in whole or in part for consideration for any other degree or qualification at this or any other university. This work is mine, except where noted in the text and Acknowledgements for collaborative efforts.

Mónica Michelle Villegas Arias
March 2026

Acknowledgements

I would like to express my sincere gratitude to the Agencia Nacional de Investigación y Desarrollo (ANID) for the financial support provided through the Beca Doctorado Nacional, which made it possible for me to pursue and complete my doctoral studies.

I also gratefully acknowledge the support of the Universidad Técnica Federico Santa María, for providing an academically stimulating environment and complementary financial support that contributed significantly to the development of this research.

Finally, I would like to thank my teachers for their guidance, knowledge, and dedication, whose contributions have been fundamental to my academic and professional development throughout my doctoral studies.

Abstract

The rapid expansion of Internet of Things (IoT) applications requires robust mechanisms to ensure the security, reliability, and maintainability of embedded software throughout its lifecycle. Over-the-Air (OTA) update systems play a central role in enabling the continuous evolution of IoT deployments. Despite their importance, OTA solutions are often designed in an ad-hoc manner, supported by fragmented guidelines that lack a structured basis for selecting mechanisms and techniques aligned with the quality needs of IoT systems. This thesis presents a consolidated catalog for designing OTA update systems in IoT environments, developed through a review of academic and industrial literature. The catalog comprises 34 techniques organized into six OTA update mechanisms, each with representative use cases and a mapping to relevant quality attributes that make beneficial and adverse impacts explicit. The catalog was evaluated through a controlled industrial experiment involving 10 engineers, balanced between novices and experts, who designed an OTA update system for a real application scenario using either their prior knowledge and experience or the catalog. This thesis offers five contributions: (1) it defines six mechanisms that structure the end-to-end update process; (2) it introduces DeOTA-IoT, a novel catalog of 34 techniques for designing OTA update systems for IoT, systematically organized into the six OTA update mechanisms defined; (3) it clarifies the notions of technique and mechanism within the OTA context, providing precise architectural definitions that have been missing from previous studies; (4) it reports an experimental validation conducted in an industrial setting, using real subjects and tasks to assess the catalog's practical usefulness; and (5) it provides a quality-attribute trade-off analysis that evaluates each technique across key attributes such as security, scalability, performance, availability, interoperability, reliability, privacy, energy management, flexibility, and evolvability, using a 5-point Likert-style bipolar scale. Together, these contributions establish a coherent foundation for systematic and quality-aware OTA update system design.

Table of contents

Nomenclature

Acronyms / Abbreviations

AES Advanced Encryption Standard

API Application Programming Interface

AUTOSAR Automotive Open System Architecture

BLE Bluetooth Low Energy

CoAP Constrained Application Protocol

COSE CBOR Object Signing and Encryption

CRC Cyclic Redundancy Check

CRUD Create, Read, Update, Delete

DTLS Datagram Transport Layer Security

E2E End-to-End

ECDSA Elliptic Curve Digital Signature Algorithm

ED25519 Edwards-curve Digital Signature Algorithm

FCM Firebase Cloud Messaging

FOTA Firmware Over-the-Air

GATT Generic Attribute Profile

gRPC gRPC Remote Procedure Calls

HSM Hardware Security Module

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IETF Internet Engineering Task Force

IoT Internet of Things

JSON JavaScript Object Notation

JWT JSON Web Token

KMS Key Management Service

LoRa Long Range

LoRaWAN Long Range Wide Area Network

MAC Message Authentication Code

MQTT Message Queuing Telemetry Transport

OS Operating System

OTA Over-the-Air

PKI Public Key Infrastructure

PSK Pre-Shared Key

RAM Random Access Memory

REST Representational State Transfer

RTO Recovery Time Objective

RTT Round-Trip Time

SD Secure Digital

SDK Software Development Kit

SE Secure Element

SLA Service Level Agreement

SOTA Software Over-the-Air

TLS Transport Layer Security

TPM Trusted Platform Module

UDP User Datagram Protocol

List of figures

List of tables

Chapter 1

Introduction

The Internet of Things (IoT) has evolved into a foundational layer of modern Cyber–Physical Systems (CPS) (Sharma et al., 2022), spanning applications ranging from mission-critical military and industrial systems to large-scale smart cities, precision agriculture, smart homes, and healthcare infrastructures (Ahmed et al., 2019).

Minerva et al. (2015) conceptualize IoT from two complementary perspectives: (1) small-scale deployments, where identifiable "Things" embed sensing, actuation, and programmability capabilities; and (2) large-scale environments, where vast populations of interconnected devices collaborate to execute complex distributed processes. Across both scales, ensuring that devices remain continuously updated is essential to avoid malfunctioning behavior, maintain operational reliability, and mitigate evolving security threats. However, in IoT deployments, updating devices is considerably challenging due to their heterogeneity, hardware and energy constraints, and intermittent connectivity, all of which require systematic, resource-aware update mechanisms (Fagan et al., 2020). As a result, IoT ecosystems require update strategies that are not only secure and reliable but also systematic, resource-aware, and adaptable to diverse deployment conditions.

Given the centrality of update processes in IoT, a variety of approaches have been developed to distribute software and firmware revisions across deployed devices. Among these, Over-the-Air (OTA) update processes have emerged as the predominant approach for sustaining the long-term evolution of IoT systems (El Jaouhari and Bouvet, 2022). OTA updates are architected, remotely orchestrated procedures that deliver, verify, and activate software or firmware revisions directly on embedded devices, eliminating the need for physical access. By enabling continuous and scalable maintenance, OTA updates address many of the inherent challenges of IoT deployments, supporting secure distribution, failure-tolerant installation, and coordinated management of large and diverse device fleets (El Jaouhari and Bouvet, 2022; Villegas et al., 2019).

Despite their central role in IoT system dependability, OTA updates are still frequently designed in an ad-hoc manner. Existing architectural guidelines remain fragmented and do not provide a clear basis for selecting solutions that align with the specific requirements and quality needs of IoT environments (El Jaouhari and Bouvet, 2022). Although previous studies have proposed taxonomies and early attempts to organize the OTA update domain (El Jaouhari and Bouvet, 2022; Moran et al., 2021; Villegas and Astudillo, 2020), the available knowledge is dispersed across heterogeneous sources and lacks integration into a coherent architectural perspective. As a result, developers often rely on isolated practices, and there is no standardized approach for incorporating security, maintainability, and other quality attributes (QA), understood as non-functional properties that characterize how a system behaves under particular conditions (Bass et al., 2021), into OTA update systems (Rajmohan et al., 2022). This is particularly problematic because OTA design decisions inherently involve trade-offs among QAs such as security, availability, energy management, and flexibility. In this work, the term "trade-off" is used to describe situations where improving one quality attribute may negatively affect another, requiring architects to balance competing design concerns (Bass et al., 2021). For instance, cryptographic verification strengthens security but increases energy consumption, while modular update strategies enhance maintainability but may introduce performance overhead (Kuppusamy et al., 2018).

To address this gap, this work consolidates the dispersed and inconsistently described body of knowledge on OTA updates into a coherent catalog of architectural techniques for Designing OTA Update Systems (DeOTA-IoT). Through a literature review and a structured unification process, the study identifies recurring solution elements and organizes them into a set of well-defined techniques, which are concrete design procedures that solve a specific OTA-related problem; and mechanisms, which group related techniques that collectively implement key stages of the OTA update process. To assess the catalog's practical relevance, a controlled industrial experiment was conducted to determine whether the explicit articulation of techniques and their associated quality-attribute impacts improves the completeness, consistency, and accuracy of architectural decisions compared to unaided, experience-based design. The rigor underlying the catalog's construction, combined with insights from the industrial study, provides an initial indication that it supports more systematic and transparent architectural decision-making for OTA-enabled update systems.

This thesis contributes to the body of knowledge in software architecture and IoT systems engineering in five ways: (1) it defines six mechanisms that structure the end-to-end update process; (2) it introduces DeOTA-IoT, a novel catalog of 34 techniques for designing OTA update systems for IoT, systematically organized into the six OTA update mechanisms defined; (3) it clarifies the notions of technique and mechanism within the OTA context,

providing precise architectural definitions that have been missing from previous studies; (4) it reports an experimental validation conducted in an industrial setting, using real subjects and tasks to assess the catalog's practical usefulness; and (5) it provides a quality-attribute trade-off analysis that evaluates each technique across key attributes such as security, scalability, performance, availability, interoperability, reliability, privacy, energy management, flexibility, and evolvability, using a 5-point Likert-style bipolar scale (Batyrshin et al., 2017; Sangster et al., 2001). By making these trade-offs explicit, the catalog consolidates dispersed knowledge and provides architects with a structured basis for systematically and balancedly designing OTA update systems across diverse IoT contexts.

The remainder of this work is structured as follows. Chapter 2 presents background information on OTA updates in IoT systems. Chapter 3 surveys related work on OTA update systems and architectural decision support. Chapter 4 presents the research method, hypotheses, and research questions adopted in this thesis. Chapter 5 introduces DeOTA-IoT, an OTA update techniques catalog for IoT systems. Chapter 6 reports the experimental validation of the proposed catalog. Chapter 7 discusses and interprets the findings derived from evaluations. Finally, Chapter 8 summarizes the contributions of this work and outlines directions for future research.

Chapter 2

OTA Updates in IoT Systems

This section presents relevant background of over-the-air (OTA) update systems in IoT environments. It introduces the fundamental concepts, architectural elements, and common challenges. The material provided in this section establishes the technical context required to understand the OTA mechanisms and techniques discussed in the subsequent chapters.

2.1 OTA Update Systems

Software evolution is a fundamental architectural concern in distributed systems. Updates enable systems to correct defects, introduce new functionality, and extend capabilities over time (Bass et al., 2021). They may target software, data, or hardware components, and can be delivered through physical interfaces or remotely across a network. In large and heterogeneous IoT environments, where devices are geographically distributed, resource constrained, and often intermittently connected, remote delivery becomes a practical necessity for maintaining system continuity.

OTA updates are commonly defined as the remote distribution and installation of software on embedded devices by transferring update artifacts from a backend system to nodes across a network (Villegas and Astudillo, 2020). Architecturally, OTA update solutions integrate two complementary subsystems (Villegas et al., 2019). The first is the embedded software responsible for executing the update logic on each device. The second is the backend infrastructure that supports update preparation, validation, orchestration, distribution, and device coordination at scale. Together, these subsystems enable a continuous and reliable update process suitable for distributed IoT deployments.

2.2 Conceptual and Architectural Elements of OTA Update Systems

This section builds on a previously published study conducted as part of this research line Villegas and Astudillo (2020), in which the core OTA update mechanisms were systematically identified and defined. In that study, a taxonomy of OTA update mechanisms was derived from a systematic review of academic and grey literature, resulting in the identification of six architectural mechanisms that capture the fundamental responsibilities of OTA update systems for IoT. Building on this foundation, the present section revisits and refines the distinction between mechanisms and techniques, clarifying their respective roles in structuring OTA update architectures and their concrete realization.

Following the terminology introduced in Villegas and Astudillo (2020), it is useful to distinguish between mechanisms and techniques when characterizing OTA update systems. A mechanism represents an architectural building block that encapsulates a specific responsibility within the update process. Mechanisms may be implemented in software, hardware, or a hybrid of both, and they define the structural and behavioral boundaries of the OTA subsystem. Examples include secure delivery, validation, installation management, scheduling, rollback control, and device orchestration. As abstractions of system functionality, mechanisms establish the interfaces through which OTA components interact and can be composed to form complete update workflows.

A technique, in contrast, denotes the concrete procedure through which a mechanism is realized. Techniques describe the operational steps used to prepare, transfer, verify, and apply updates across distributed IoT devices. They reflect engineering practices grounded in research, prevailing standards, and accumulated industry experience, and are typically adapted to the constraints of particular hardware platforms, protocols, and deployment environments. While mechanisms articulate the architectural responsibilities within an OTA system, techniques specify how these responsibilities are executed in practice.

Techniques operate at the implementation level and address operational concerns such as reliability, efficiency, scalability, and security. Each technique represents a tangible procedure with its own workflow and measurable effects on system qualities. Together, mechanisms and techniques provide a structured way to interpret the functional and operational layers of OTA update systems and to reason about the interplay between architectural responsibilities and their concrete realization.

In addition to mechanisms and techniques, architectural knowledge in software engineering is often expressed through patterns and tactics, which operate at different abstraction levels (Harrison and Avgeriou, 2010). Architectural patterns describe reusable structural

solutions that organize system components and their interactions at a high-level, addressing recurring design problems across domains (Buschmann et al., 1996). Architectural tactics, in contrast, represent low-level design decisions intended to influence specific quality attributes, such as security, performance, or availability (Bass et al., 2021).

While OTA update mechanisms define architectural responsibilities within the update process and techniques specify concrete operational procedures for realizing those responsibilities, patterns and tactics play a complementary role: patterns may encompass multiple mechanisms as part of a broader architectural solution, whereas tactics are often instantiated within or across techniques to achieve targeted quality-attribute effects.

Building on this distinction, in Villegas and Astudillo (2020) was proposed a taxonomy that organizes OTA update systems into six mechanisms, each capturing a core architectural responsibility. As summarized in Figure 2.1, these mechanisms cover secure update preparation, management, dissemination and installation, recovery, scheduling, and packaging, providing a clear structural view of the OTA update process.

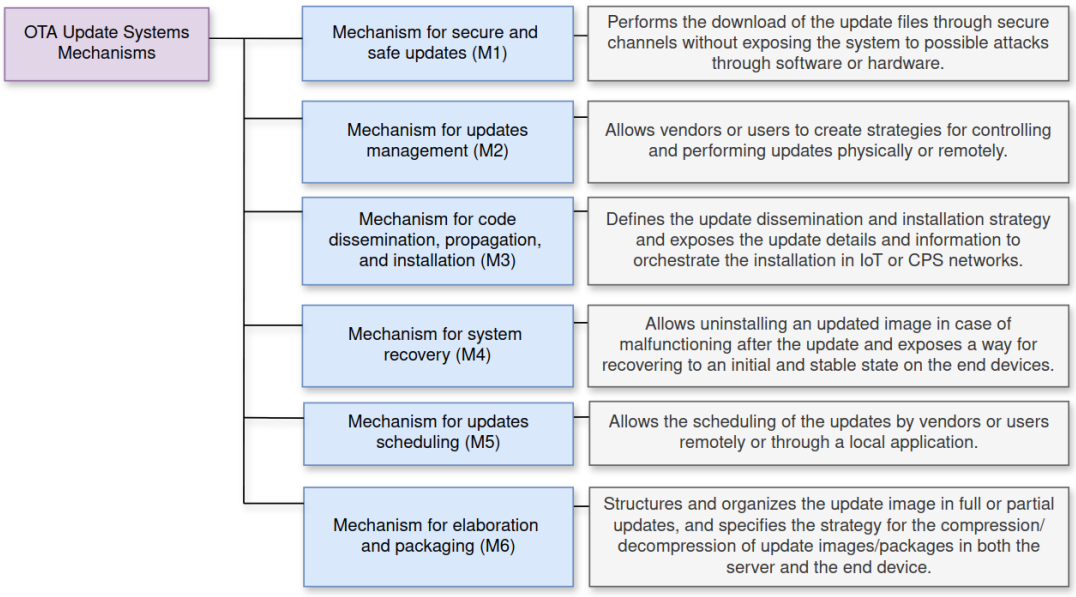


Fig. 2.1 Description of the six mechanisms that set up an OTA update system for IoT, created according to the definition proposed by Villegas and Astudillo (2020).

As shown in Figure 2.1, the taxonomy structures OTA update systems around six architectural mechanisms that capture the main responsibilities of the update process, including secure preparation, lifecycle management, dissemination and installation, recovery, scheduling, and packaging. This classification offers a clear architectural decomposition of the OTA

subsystem and clarifies how these responsibilities contribute to a coherent end-to-end update workflow.

While the taxonomy clarifies the structural organization of OTA responsibilities, understanding how these responsibilities unfold during an update requires examining their dynamic interaction. A runtime perspective exposes how mechanisms coordinate to advance the update process, how control and data flows progress through preparation, distribution, verification, installation, and recovery, and how the techniques associated with each mechanism are invoked to realize these transitions in practice.

As an illustrative example, Figure 2.2 expands and refines the update sequence discussed in Villegas et al. (2019), providing a more detailed view of how preparation, distribution, verification, installation, and recovery activities relate to one another. The figure specifies the connections between the steps of the OTA process, the techniques that operationalize each step (shown in the highlighted elements), and the mechanisms they instantiate, represented through the connecting structure.

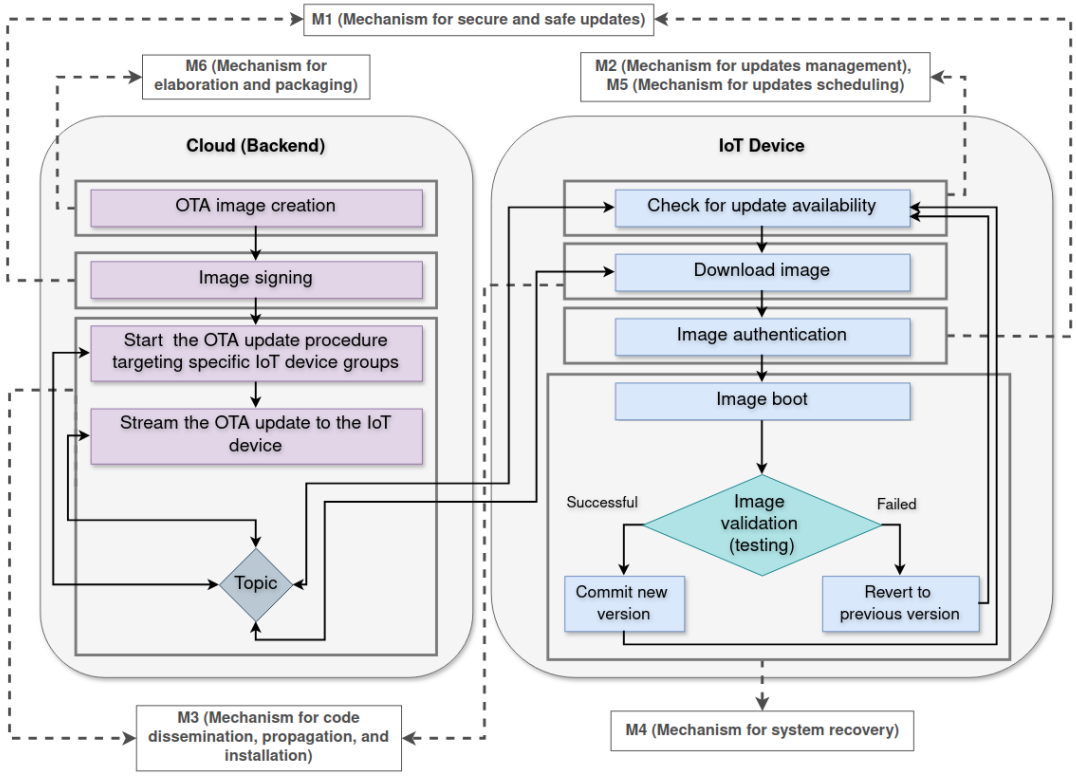


Fig. 2.2 Example of an OTA Update System for IoT, in which the techniques are inside the highlighted elements, and the mechanisms they are associated with are shown as connected.

As illustrated in Figure 2.2, the OTA update process is structured as a coordinated workflow that spans both the backend and the IoT device. The figure outlines the progression from image creation and signing, through distribution and verification, to installation and validation, making explicit the main control flows that govern the update lifecycle. It also shows how specific techniques instantiate the mechanisms responsible for secure delivery, update management, dissemination, and recovery. By exposing these relationships in a unified view, the diagram provides a clear architectural reference that supports reasoning about the integrity, safety, and robustness of the OTA process across heterogeneous IoT devices.

Chapter 3

Related Work

This chapter surveys the related work relevant to the research presented in this thesis and is structured into two sections. The first section reviews existing studies related to OTA update mechanisms, focusing on architectural abstractions, responsibilities, and system-level approaches proposed for structuring OTA update processes. The second section examines related work on OTA update techniques, including approaches for organizing, classifying, and evaluating concrete OTA update procedures. Together, these sections position the contributions of this thesis within the existing body of knowledge and highlight the gaps addressed by the proposed catalog.

3.1 Related Work on OTA Update Mechanisms

Some efforts have proposed taxonomies to organize OTA update solutions, primarily to structure design concerns or classify existing approaches. While these efforts provide useful insights into specific aspects of OTA updates, they remain limited in terms of architectural scope and level of abstraction.

Brown and Sreenan Brown and Sreenan (2013) surveyed software update features in wireless sensor networks and proposed a taxonomy organized around design decisions. Their taxonomy comprises five categories: update dissemination, update activation, fault detection, fault recovery, and management. While this work represents an early attempt to structure OTA-related concerns, its categories largely reflect functional stages rather than explicit architectural responsibilities. As such, the taxonomy does not formalize mechanisms as reusable architectural building blocks, nor does it provide guidance on how these categories interact or compose within an end-to-end OTA update architecture.

Halder et al. Halder et al. (2019) surveyed remote OTA software updates in the automotive domain, with a strong emphasis on security aspects. They proposed a taxonomy of secure

OTA update techniques organized into seven categories: cryptographic primitives, blockchain-based approaches, secure hardware modules, and secure update frameworks. Although this taxonomy provides a detailed view of security-related solutions, it is narrowly scoped to security mechanisms and cryptographic techniques. Other architectural responsibilities,such as update management, dissemination strategies, scheduling, recovery, installation control, and packaging, are not addressed, limiting its applicability as a comprehensive architectural framework for OTA update systems.

Standardization-oriented efforts also contribute to the structuring of OTA update systems. The U.S. National Telecommunications and Information Administration (NTIA), through its IoT Security Upgradability Existing Standards, Tools and Initiatives Working Group, compiled a catalog of existing IoT security standards National Telecommunications and Information Administration (NTIA) (2017). This catalog includes frameworks such as The Update Framework (TUF) Asokan et al. (2018), which focuses on securing software update pipelines through role separation, metadata signing, and compromise resilience. While these initiatives provide valuable normative guidance and best practices, they primarily target security assurance and interoperability. They do not aim to decompose the OTA update process into architectural mechanisms, nor do they offer a systematic classification of OTA responsibilities beyond security-focused concerns.

Across these taxonomies and catalogs, a common limitation emerges. Existing works either focus on specific quality attributes, most notably security, or organize OTA knowledge around functional stages or implementation techniques. None of the reviewed approaches provides an explicit architectural decomposition of the OTA update process into a set of well-defined, technology-agnostic mechanisms that capture the core responsibilities of OTA systems across domains and deployment contexts.

In contrast, the approach adopted in this thesis builds on a previously published study conducted as part of this research line Villegas and Astudillo (2020), which analyzes the OTA update process holistically and identifies a set of core OTA update mechanisms as architectural abstractions. These mechanisms are defined independently of specific technologies or protocols and are intended to structure the end-to-end update process at an architectural level. By explicitly distinguishing mechanisms from techniques, this work provides a foundation for systematically organizing concrete OTA update procedures and reasoning about their quality-attribute implications, addressing limitations observed in prior taxonomies. Figure 3.1 shows a contributions timeline of the mentioned references.

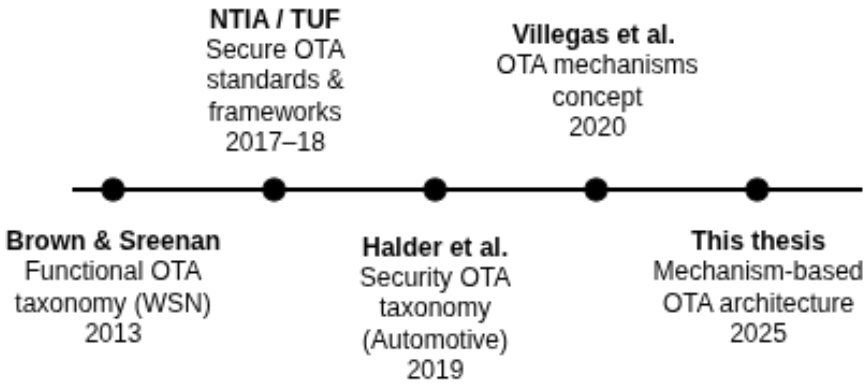


Fig. 3.1 Contributions timeline of the related work on OTA Update Mechanisms.

3.2 Related Work on OTA Update Techniques Catalogs

Research on OTA update systems for IoT has advanced across several technical domains, including secure firmware delivery, dissemination efficiency, protocol and platform support, and architectural guidance. These efforts underscore the importance of OTA processes for maintaining the long-term reliability, security, and maintainability of connected devices, as highlighted by Bauwens et al. (2020), Bradley and Barrera (2023), and Hernández-Ramos et al. (2020). However, despite these contributions, existing work provides limited support for practitioners who require systematic, architecture-oriented methods for designing OTA update systems that must operate across heterogeneous devices, networks, and quality requirements.

A broad stream of research has concentrated on strengthening the security of OTA transmission channels. Shin and Jeon (2024) propose an OTA protocol that integrates MQTT (Johnson, 2025) with Merkle trees to enhance integrity protection during firmware distribution. Similarly, Chien and Wang (2022) present an MQTT 5.0-based OTA architecture emphasizing the risks posed by insecure update pipelines to entire IoT ecosystems. In low-power wide-area contexts, Hayati (2024) explores Firmware Updates Over-the-Air (FUOTA) procedures for Long Range Wide Area Network (LoRaWAN) devices (Lau et al., 2025). Park et al. (2025) further contribute a hybrid cryptographic update mechanism that combines symmetric and asymmetric primitives to safeguard firmware authenticity and confidentiality in constrained IoT devices. These studies demonstrate that secure communication channels and integrity mechanisms mitigate individual threats, but they remain confined to protocol-level or cryptographic concerns and do not provide broader architectural guidance for systematically composing update responsibilities.

A second body of work addresses efficiency and dissemination challenges in constrained environments. Malumbres et al. (2024) propose hybrid unicast–broadcast dissemination

strategies to reduce latency in low-end or hard-to-access devices. Mahfoudhi et al. (2022) analyze update procedures for Narrowband IoT (NB-IoT) devices (Heins, 2022), to minimize energy and bandwidth consumption during OTA operations. Although these contributions provide valuable optimizations for specific device types and network technologies, they do not generalize into an architectural framework capable of supporting systematic decision-making across heterogeneous IoT deployments.

Complementary efforts introduce architectural abstractions and best practices. Sousa and Borin (2023) present a layered OTA model emphasizing the integration of security and privacy across architectural levels. Anedda et al. (2023) discuss best practices for secure and privacy-aware IoT system design, underscoring the importance of disciplined engineering approaches. Yet, these works remain high-level: they neither consolidate OTA techniques nor provide a systematic method for evaluating or selecting them according to diverse quality attributes.

Standardization efforts such as RFC 9019 (Moran et al., 2021) defines a normative firmware update architecture including manifests, bootloaders, firmware consumers, and recovery elements, along with essential security and interoperability requirements. While these specifications establish foundational architectural principles, they intentionally abstain from detailing concrete techniques, implementation variability, or architectural trade-offs related to attributes such as reliability, scalability, or energy management.

Several surveys and domain analyses offer broader syntheses. Arakadakis et al. (2021) examine Over-the-Air Programming (OTAP) techniques for heterogeneous and resource-constrained IoT nodes, discussing protocol-level dissemination strategies and performance constraints. Kesari et al. (2025) provide an overview of OTA concepts across Cloud, edge, and hybrid environments, including Software Over-the-Air (SOTA), Firmware Over-the-Air (FOTA), rollback strategies, secure boot, and delta updates. El Jaouhari and Bouvet (2022) focus specifically on secure firmware OTA methods, reviewing threat landscapes, reference architectures, and standardization initiatives. These surveys map the technological landscape but remain primarily descriptive and do not structure OTA knowledge into an architectural framework that relates techniques to quality-attribute trade-offs.

A more architecturally aligned perspective is offered in Villegas et al. (2019), in which IoT, embedded, and CPS (Sharma et al., 2022) were analyzed to assess their support for OTA updates. Such a study identifies platforms that incorporate OTA capabilities, but does not extract reusable architectural guidance or a design-oriented catalog of techniques. Building on this, Villegas and Astudillo (2020) introduced a preliminary taxonomy of OTA mechanisms and linked several techniques to each mechanism. While this taxonomy constitutes an important first step, it lacks detailed technique descriptions, fails to characterize the quality-

attribute implications of techniques, and provides no systematic guidance for selecting or combining techniques during OTA system design.

Across these research strands, a clear limitation persists: none of the existing studies provides a systematic, architecture-oriented framework that organizes OTA mechanisms and techniques and articulates their associated quality-attribute trade-offs. Existing work addresses isolated concerns but does not support architects in reasoning about how to select, combine, or adapt OTA techniques to meet diverse IoT requirements such as security, availability, energy management, and scalability. Figure 3.2 shows a contributions timeline of the mentioned references.

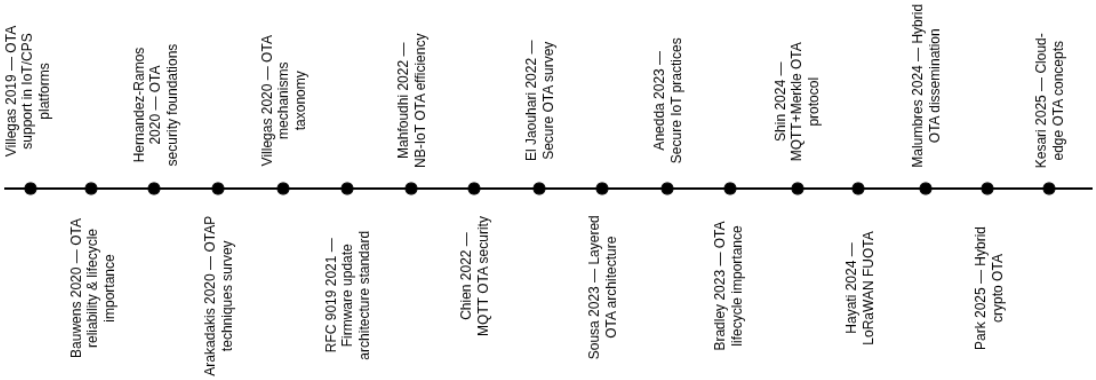


Fig. 3.2 Contributions timeline of the related work on OTA Update Techniques Catalogs.

Chapter 4

Methods and Materials

4.1 Research Scope and Study Overview

This chapter presents the methods and materials used in the research reported in this thesis. The thesis builds on two sequential, conceptually connected studies that together provide the methodological foundation to identify, extract, consolidate, and refine the techniques that compose the DeOTA-IoT catalog. The goal was to build a comprehensive, coherent, and technically grounded inventory of OTA update techniques for IoT systems while enabling a systematic comparison of their trade-offs across key QAs.

The first study focuses on identifying and defining OTA update mechanisms. This study aimed at structuring the OTA update design space at an architectural level. Through a systematic analysis of academic and grey literature, OTA update systems were decomposed into a set of core architectural mechanisms, each representing a distinct responsibility within the end-to-end update process. The resulting taxonomy provides a stable and technology-agnostic foundation for reasoning about OTA update mechanisms. This study follows an exploratory and consolidative methodological approach, oriented toward identifying recurring architectural responsibilities rather than exhaustively cataloging implementation-level solutions.

The second study, which constitutes the main contribution of this thesis, builds directly on this foundation by analyzing, organizing, and evaluating OTA update techniques. Techniques are studied as concrete procedures that realize the previously defined mechanisms and operationalize their responsibilities under specific technological and deployment constraints.

This work introduces a structured techniques catalog, incorporates quality-attribute-driven trade-off analysis, and evaluates the catalog through both quantitative experimentation and qualitative practitioner feedback. In contrast to the first study, this study adopts a

systematic and evaluative methodology, supported by explicit search, filtering, and assessment procedures.

Figure 4.1 illustrates the overall workflow following recognized guidelines for evidence-based software engineering (Kitchenham et al., 2009) and PRISMA recommendations (PRISMA, 2024). It consists of eleven well-defined steps explained in three main stages: (i) literature identification and screening, (ii) extraction and homologation of techniques, and (iii) integration and consolidation of results from both the previous and current studies. Each step illustrated in Figure 4.1 is described in detail in the corresponding subsections of this section, following the order of the workflow.

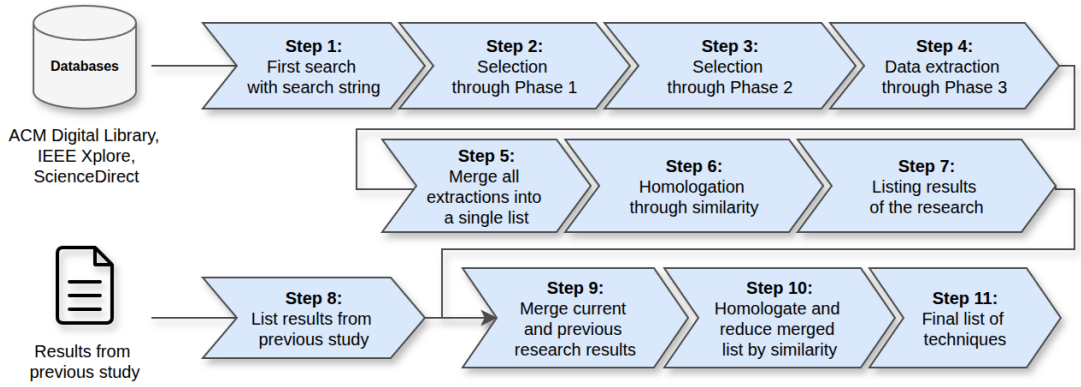


Fig. 4.1 Overview of the research workflow combining new evidence from digital libraries with results from the previous study (Villegas and Astudillo, 2020).

In Figure 4.1, Steps 1–3 correspond to the identification and screening of relevant studies (Section 4.3.1). Step 4 extracts OTA-related techniques, while Steps 5–7 consolidate and homologate the extracted data into a unified list (Section 4.3.2). Steps 8–10 integrate results from the previous study with the new evidence and apply a second homologation pass (Section 4.3.3). Step 11 produces the final consolidated set of techniques used to construct the DeOTA-IoT catalog

By explicitly separating the architectural definition of mechanisms from the systematic study of techniques, this chapter clarifies the scope, dependencies, and progression of the research. The mechanisms taxonomy establishes the architectural context, while the techniques catalog represents the design artifact evaluated in this thesis. Together, these studies support a systematic, quality-aware approach to architecting OTA update systems for IoT environments.

4.2 Research Requirements

This section introduces the research questions and methodological requirements that guided the studies presented in this thesis. The research comprises two sequential and complementary studies: (i) a study focused on the identification and definition of OTA update mechanisms at the architectural level, and (ii) a study centered on the systematic construction and evaluation of a structured OTA update techniques catalog (DeOTA-IoT). While both studies address OTA update systems for IoT, they differ in scope and methodological emphasis. Accordingly, the requirements presented in this section apply at a high level to both studies, whereas the detailed search and filtering procedures described in the following subsections are primarily associated with the techniques catalog study. The methodological procedure adopted for the mechanisms study is summarized separately in Section 4.4.

4.2.1 Research Hypotheses

The hypotheses guiding the research are:

- H_0: The use of the catalog does not have a significant impact on the quality or efficiency of OTA system design.
- H_1: The use of the catalog significantly enhances the quality and efficiency of OTA system design, leading to more systematic decisions that align with key quality attributes such as security, maintainability, and interoperability.

4.2.2 Research Questions

Two research questions guided the investigation and its sequential studies:

- **RQ0:** How can OTA update mechanisms and techniques be systematically organized to support a clear and useful catalog for system designers?
- **RQ1:** How do OTA update techniques influence critical IoT quality attributes such as security, reliability, availability, scalability, and energy management?

To answer these questions, the research adopted the following methodological actions:

1. **Mechanism definition and technique organization (addresses RQ0):** The research first analyzed OTA update systems reported in academic and professional sources to identify recurring architectural responsibilities involved in the end-to-end update process. These responsibilities were consolidated into a set of OTA update mechanisms,

providing a stable architectural structure. Building on this foundation, a systematic study of OTA update techniques was conducted, in which concrete procedures reported in the literature were extracted, merged, and homologated. Each technique was then mapped to one or more mechanisms according to the responsibility it realizes, resulting in a structured techniques catalog.

2. **Quality attribute impact assessment (addresses RQ1):** OTA update techniques were evaluated with respect to their impact on selected IoT quality attributes. This assessment was conducted using expert judgment and a 5-point bipolar Likert scale, allowing both positive and negative effects of each technique on relevant quality attributes to be explicitly represented.

4.2.3 Search Strategy

This subsection describes the search strategy adopted for the systematic identification of OTA update techniques used to build the DeOTA-IoT catalog.

The literature search was designed to capture a broad range of primary studies describing OTA mechanisms, processes, and technical components for IoT systems. The search was conducted across the main digital libraries commonly used in software engineering and IoT research, as listed in Table 4.1.

Table 4.1 Digital libraries queried during the literature search.

Name	URL
ACM Digital Library	https://dl.acm.org
IEEE Xplore	https://ieeexplore.ieee.org
ScienceDirect	https://www.sciencedirect.com

To retrieve studies addressing OTA updates in IoT systems, the following search string was used:

```
(OTA OR over-the-air) AND (IoT OR "internet of things")
```

To select relevant material and ensure methodological rigor, inclusion and exclusion criteria were defined and consistently applied during the screening process:

- **Exclusion Criteria:**
  - **EC1.** Texts not written in English
  - **EC2.** Not fully available texts

  - **EC3.** Patents
  - **EC4.** Not accessible texts
  - **EC5.** Presentations

- **Inclusion Criteria:**
  - **IC1.** A primary study must contain software or hardware components used in the development of OTA update systems.
  - **IC2.** A primary study must report software or hardware techniques, approaches, or methods for designing or implementing an OTA update process.

ACM Digital Library, IEEE Xplore, and ScienceDirect were selected because they collectively cover the primary publication venues in software engineering, embedded systems, and IoT research, ensuring broad coverage of both architectural and implementation-oriented OTA update studies. ArXiv was not used because preprints were not included, ensuring that only peer-reviewed and stable publications were considered in the evidence base.

The search string was constructed following PICOC (Kitchenham and Charters, 2007) guidelines (Population, Intervention, Comparison, Outcome, Context). Population corresponds to IoT systems; Intervention to OTA update mechanisms or techniques; Outcome to OTA update design or implementation approaches; and Context to distributed or embedded IoT environments.

4.2.4 Study Filtering Phases

This subsection describes the filtering phases applied to the literature collected for the OTA update techniques catalog study.

To ensure methodological rigor and relevance, inclusion and exclusion criteria were defined and consistently applied during the screening process. These criteria were used to identify studies that explicitly reported software or hardware techniques applicable to the design or implementation of OTA update systems.

- **Phase 1 (PH1):** Selection of articles in which the title and abstract were related to techniques, approaches, or methods for designing or implementing OTA update systems for IoT.
- **Phase 2 (PH2):** Selection of articles in which the introduction and proposal were related to methods, techniques, processes or approaches in the context of designing or implementing OTA update systems for IoT.

- **Phase 3 (PH3):** Extraction of the techniques used in the OTA update systems reported for IoT. The extraction method involves obtaining the technical components, either software or hardware components involved in the design or implementation process of the OTA Update system.

The search, screening, and filtering phases were conducted by the primary researcher. Inclusion and exclusion decisions followed predefined criteria to ensure consistency and reproducibility.

4.3 Research Process

This section describes the research process adopted for the second study of this thesis, which focuses on the construction and consolidation of the DeOTA-IoT techniques catalog. This study builds on the architectural foundation established by the prior identification of OTA update mechanisms, but adopts a distinct and systematic process aimed at aggregating, homologating, and refining OTA update techniques reported in the literature.

The process extends earlier evidence by incorporating five additional years of research (2020–2025) and applies a structured, multi-step methodology to produce a unified, comprehensive, and architecturally meaningful catalog of OTA update techniques. The methodological procedure used to identify and define OTA update mechanisms is described separately and is not repeated here.

4.3.1 Literature Identification and Screening

Step 1 executed the multi-library search. *Steps 2 and 3* applied PH1 and PH2 filters, ensuring that only studies directly relevant to OTA updates in IoT were retained.

Figure 4.2 illustrates the number of studies identified, screened, and included after applying the filtering phases.

4.3.2 Technique Extraction and Initial Homologation

Step 4 extracted OTA-related techniques based on PH3. *Step 5* merged all extracted data into a unified dataset. *Step 6* conducted the first homologation pass, consolidating techniques exhibiting functional equivalence, similar operational goals, and differences limited to naming or implementation context. *Step 7* produced a structured list of techniques representing the updated evidence base for 2020–2025.

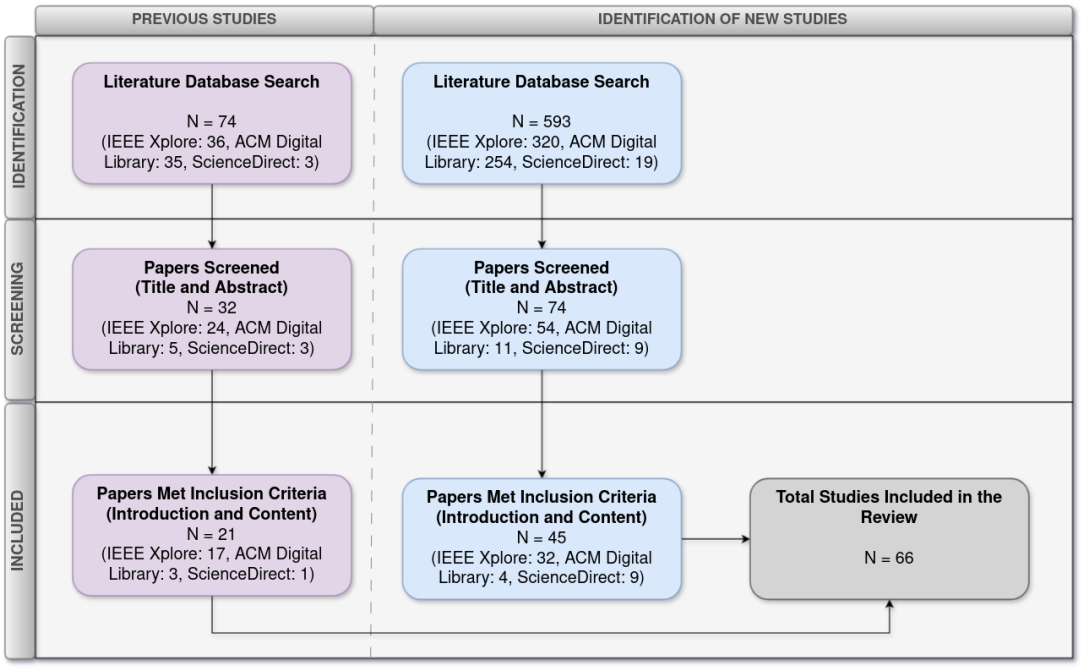


Fig. 4.2 Study identification, screening, and inclusion across the different digital libraries.

4.3.3 Integration with Previous Evidence and Final Consolidation

Step 8 reorganized techniques identified in the previous study (Villegas and Astudillo, 2020). *Step 9* integrated prior and new datasets. *Step 10* applied a second homologation pass to eliminate deep redundancies and ensure conceptual coherence across both periods. *Step 11* produced the final set of 34 techniques after a rigorous multi-stage consolidation process.

To transform this heterogeneous collection into a coherent and architecturally meaningful catalog, two consecutive homologation passes were applied. First, techniques were consolidated exhibiting functional equivalence across the reviewed sources (Charalampidis et al., 2022; Teck Khieng et al., 2023; Da Silva et al., 2023; Srivastava et al., 2021; Zyrianoff et al., 2024; Witanto et al., 2019; Thantharate et al., 2019; Ali et al., 2025; Srinivas et al., 2022; Sun et al., 2021; Arakadakis et al., 2022; Gu et al., 2024; K et al., 2024; Krishnan et al., 2024; Arakadakis et al., 2021; Sun et al., 2023; Abdelfadeel et al., 2020; Jang and Tsai, 2021; Wei et al., 2025; Catalano, 2021; Kubaščík et al., 2024; Zhang et al., 2025; Sudharsan et al., 2022; Nguyen et al., 2024; de Sousa et al., 2022; Nikbakht Bideh and Gehrmann, 2022; Andrade et al., 2019; El Jaouhari and Bouvet, 2022; Kerliu et al., 2019; Crowther et al., 2022; He et al., 2019; Jaouhari and Bouvet, 2022; El Jaouhari, 2022; Thakur et al., 2019; Aoki et al., 2024). Then, the resulting set was refined by grouping techniques according to their architectural role and intended objective within the OTA update process. Redundant, overlapping, or

overly specialized entries were merged into unified, semantically consistent techniques that preserved their essential functional intent.

The result of this refinement is a distilled set of 34 techniques, each representing a clear, well-defined, and architecturally significant approach within the DeOTA-IoT catalog.

For traceability, each technique in the consolidated catalog was linked to the primary studies from which it was derived, thereby enabling verification of the search, selection, and homologation decisions and supporting the study's replicability.

4.4 Methodological Procedures of the Studies

This research comprises two sequential studies with distinct but complementary methodological procedures. The first study focuses on the identification and definition of OTA update mechanisms at the architectural level, while the second study builds upon this foundation to construct and evaluate an OTA update techniques catalog. Although both studies address OTA update systems for IoT, they differ in scope, objectives, and methodological emphasis. The methodological procedures followed in each study are summarized below.

4.4.1 Methodological Procedure for the Mechanisms Study

The objective of the first study was to identify and define the core architectural mechanisms that structure the end-to-end OTA update process in IoT systems. This study adopted an exploratory and consolidative methodological approach, oriented toward identifying recurring architectural responsibilities rather than exhaustively cataloging implementation-level solutions.

The procedure involved a broad analysis of academic literature and professional sources describing OTA update systems, design recommendations, and update processes. The inclusion of professional and institutional sources was motivated by the absence of standardized OTA architectures and the significant role of practitioner-driven knowledge in this domain.

Relevant sources were analyzed to extract recurring responsibilities, functional roles, and structural elements involved in OTA update systems. These elements were compared and consolidated through a homologation process, resulting in a set of architectural mechanisms. Each mechanism represents a distinct responsibility within the OTA update process and abstracts away implementation-specific details. The outcome of this study is a taxonomy of OTA update mechanisms that provides a stable architectural framework for subsequent analysis.

4.4.2 Methodological Procedure for the Techniques Catalog Study

The second study focuses on the systematic identification, organization, and evaluation of OTA update techniques. This work builds directly on the mechanisms defined in the first study, treating them as architectural containers for organizing concrete implementation procedures.

A systematic literature review was conducted to identify OTA update techniques reported in academic studies. Techniques were extracted based on their relevance to the OTA update process and filtered according to predefined inclusion and exclusion criteria. Identified techniques were analyzed, merged, and homologated to remove redundancy and ensure consistent granularity.

Each technique was subsequently mapped to one or more OTA update mechanisms according to the responsibility it realizes within the update process. To support quality-aware decision-making, techniques were further evaluated with respect to their impact on selected IoT quality attributes using a structured trade-off representation. This evaluation was supported by expert judgment and formed the basis for the construction of the DeOTA-IoT techniques catalog.

Finally, the resulting catalog was validated through a quantitative experimental study and complemented by qualitative practitioner feedback to assess its practical utility and decision-support effectiveness.

4.5 Quality Attributes Identification and Selection

This section describes the identification and selection of quality attributes used to evaluate OTA update techniques in the DeOTA-IoT catalog study.

4.5.1 Requirements for Quality Attribute Selection

Evaluating the architectural impact of OTA update techniques requires a set of quality attributes that is applicable across heterogeneous IoT systems and independent of specific application domains. Since OTA mechanisms are employed in diverse contexts, including industrial automation, healthcare, smart environments, and large-scale distributed deployments, the selected quality attributes must capture cross-cutting concerns that influence architectural decisions beyond functional correctness.

In this work, quality attributes are understood as measurable or assessable system properties that reflect how well a system satisfies stakeholder concerns during architectural evaluation (Bass et al., 2021). Furthermore, the selected attributes must be compatible with

established software quality standards, such as ISO/IEC 25010 (ISO/IEC, 2023), while also addressing IoT-specific concerns such as energy management, interoperability, and evolvability.

4.5.2 Quality Attribute Models and Evaluation Frameworks for IoT Systems

To identify a suitable set of quality attributes, an extensive review of existing proposals for IoT system quality was conducted. Prior work has proposed numerous models, frameworks, and attribute sets aimed at evaluating IoT systems; however, these efforts vary considerably in scope, abstraction level, and domain applicability.

Several studies focus on domain-specific IoT applications. For example, Shi et al. (2013) proposed behavioral patterns to improve quality of service in mote-based IoT systems through structured code design. In environmental monitoring contexts, Kozlowski et al. (2023) applied ISO/IEC 25010 to assess smart sensing platforms. In healthcare, Chhiba et al. (2022) introduced a quality-driven reference platform tailored to medical IoT architectures. Similarly, Mishra et al. (2024) examined agricultural IoT systems, mapping quality attributes such as scalability, flexibility, and interoperability to architectural components, while Temkar and Bhaskar (2021) adapted ISO/IEC 25010 for home automation scenarios, accounting for wireless mobility and hardware–software interaction. Other work, such as Banijamali et al. (2020), identified quality drivers including scalability, timeliness, and security in IoT cloud architectures.

In addition to domain-specific approaches, several studies proposed generalized quality models and evaluation frameworks for IoT systems. Kim (2016) extended ISO 9126 (ISO/IEC, 2001) to define a quality model for IoT applications, although the model primarily addresses general service characteristics and does not explicitly consider cross-domain adaptability. Bures et al. (2020) combined existing IoT quality characteristics into a unified framework emphasizing security, privacy, reliability, and usability, mainly for testing purposes. Other proposals, such as those by Abdallah et al. (2019) and Arakaki et al. (2020), applied ISO/IEC 25010 attributes to heterogeneous IoT systems, targeting improvements in availability and maintainability in contexts such as smart homes. In industrial and automotive domains, Antonino et al. (2022) and Banijamali et al. (2019) adapted ISO/IEC 25010 to address quality concerns specific to Industry 4.0 and cloud-based automotive platforms.

Broader surveys reinforce these observations. Cadavid et al. (2020) highlighted inconsistencies in how quality attributes are treated across System-of-Systems architectures, while Washizaki et al. (2020) reviewed more than 140 IoT design and architecture patterns and

found that most are highly domain-specific and lack uniform treatment of quality concerns such as compatibility, security, and maintainability.

Overall, the reviewed literature indicates that most existing proposals define quality attributes within restricted application domains or tailor standard quality models to specific IoT contexts. Consequently, these approaches provide limited support for evaluating architectural decisions that must generalize across heterogeneous IoT systems.

4.5.3 Comparative Analysis of Quality Attribute Sets Identified in IoT Literature

The diversity and domain dependence of existing proposals motivated a comparative analysis of IoT quality attribute sets reported in the literature. Tables 4.2, 4.3, and 4.4 summarize representative proposals, highlighting their scope and limitations as well as their respective application contexts.

Based on this analysis, the set of IoT quality attributes proposed by Khezemi et al. (2024) was selected for evaluating OTA update techniques in this study. Unlike the previously discussed approaches, Khezemi et al. (2024) explicitly address software quality requirements that should be considered during the architectural design phase of IoT systems, independent of application domain. The authors conducted a systematic literature review of 103 primary studies following the PRISMA methodology (Page et al., 2021), examining how IoT quality requirements relate to architectural styles. As a result, they identified ten core quality attributes: security, scalability, performance, availability, interoperability, reliability, privacy, energy management, flexibility, and evolvability. In addition, the study analyzed how different architectural styles support these attributes across diverse IoT systems.

This domain-independent and architecture-oriented perspective aligns well with the goals of the DeOTA-IoT catalog, which aims to evaluate OTA update techniques as architectural design decisions applicable to a wide range of IoT contexts. For this reason, the quality attribute set proposed by Khezemi et al. (2024) was adopted as the basis for the analysis presented in this work.

Table 4.2 QAs Proposals Comparison (Part 1).

Authors	Application Domain	Type of Proposal	Scope	ISO Based	Main Limitations
Shi et al. (2013)	Mote-based IoT applications	Design patterns / behavioral patterns	Domain Specific	No	Focused on Quality of Service in mote-based scenarios; does not provide a reusable QA set across IoT domains.
Kozlowski et al. (2023)	Smart environmental monitoring	Quality model for a specific system type	Domain Specific	Yes (ISO/IEC 25010)	Tailored to environmental monitoring; does not generalize QAs beyond this context.
Chhiba et al. (2022)	Healthcare IoT	Reference architecture / platform	Domain Specific	Partially	Quality attributes are defined for healthcare scenarios; limited transferability to other IoT domains.
Mishra et al. (2024)	Agriculture IoT	Architectural framework	Domain Specific	Partially	Concentrates on agricultural systems; QA set and mappings are not validated across other IoT applications.
Temkar and Bhaskar (2021)	Smart home	Quality model adaptation	Domain Specific	Yes (ISO/IEC 25010)	Considers mobility and Hardware/Software interaction in home automation; does not provide a domain-independent QA catalog.

4.6 Trade-off Identification and Impact Representation

This subsection describes the process for identifying and characterizing the architectural trade-offs associated with each OTA technique. It explains how quality-attribute impacts were elicited, validated, and consolidated into a comparative model supporting RQ1.

4.6.1 Assessment Method

Trade-offs were evaluated using a 5-point bipolar scale (Sangster et al., 2001; Batyrshin et al., 2017), grounded in the formal theory of bipolar measurement. This type of scale enforces symmetry between negative and positive categories, with zero representing a meaningful

Table 4.3 QAs Proposals Comparison (Part 2).

Authors	Application Domain	Type of Proposal	Scope	ISO Based	Main Limitations
Banijamali et al. (2020)	IoT cloud architectures	Quality drivers	Domain Specific	No	Identifies key drivers (e.g., timeliness, scalability, security) for cloud-based IoT, but not as a general QA framework.
Kim (2016)	Generic IoT services	Quality model	Multi domain (service oriented)	Yes (ISO/IEC 9126)	Focused on IoT service characteristics; limited emphasis on architectural evaluation across heterogeneous IoT systems.
Bures et al. (2020)	IoT testing	Quality framework	Multi domain	Partially	Emphasizes testing and selected QAs (security, privacy, reliability, usability); does not aim at a comprehensive QA set for architecture design.
Abdallah et al. (2019)	Smart home	Evaluation model	Domain Specific	Yes (ISO/IEC 25010)	Focuses on availability and maintainability in smart homes; it lacks cross-domain generalization.
Arakaki et al. (2020)	Generic IoT systems	Evaluation framework	Domain Specific	Yes (ISO/IEC 25010)	Addresses specific heterogeneous scenarios; QA set is not framed as a reusable catalog.

and cognitively stable midpoint. As shown by Batyrshin et al. (2017), equidistant categories and a coherent numerical structure are essential for capturing direction-sensitive judgments without distorting their interpretation.

The selection of the scale was made after an analysis and review of various proposals as shown in Table 4.6. The two articles on bipolar rating scales were selected for their direct methodological relevance to the structure and interpretation of bipolar scales. Unlike general studies on Likert formats or software architecture evaluation, these references offer a rigorous and domain-independent basis for understanding how respondents process evaluations that range from negative to positive, with a clear conceptual midpoint. Since this study assesses whether a technique improves, degrades, or has no effect on a QA, the rating process must

Table 4.4 QAs Proposals Comparison (Part 3).

Authors	Application Domain	Type of Proposal	Scope	ISO Based	Main Limitations
Antonino et al. (2022)	Industry 4.0	Quality model adaptation	Domain specific	Yes (ISO/IEC 25010)	Tailored to industrial automation; concentrates on industrial needs rather than generic IoT.
Banijamali et al. (2019)	Automotive cloud platforms	Architecture and QAs	Domain specific	No	Emphasizes availability, reliability, and security in automotive systems; not intended as a generic QA basis.
Cadavid et al. (2020)	System-of-Systems	Architecture review	Cross-domain (SoS, not IoT-specific)	No	Identifies inconsistency in QA treatment, but does not define a concrete QA catalog for IoT.
Washizaki et al. (2020)	IoT design and architecture patterns	Pattern catalog	Multi-domain but pattern-oriented	No	Shows that patterns are domain-specific and often neglect QAs; does not provide a unified QA set.
Khezemi et al. (2024)	IoT systems (cross-domain)	Systematic QA catalog	Domain independent	Yes (aligned with ISO/IEC 25010)	Provides a generalized set of ten key QAs and analyzes their relationship with architectural styles across heterogeneous IoT systems.

represent a symmetric continuity where negative and positive effects are equally weighted and easy to interpret. The selected studies address this need by clarifying how bidirectional scales function, assigning meaningful numerical values, and demonstrating how centered scales contribute to valid interpretation. Batyrshin et al. (2017) provide the theoretical foundation, and Sangster et al. (2001), empirically validate the use of negative-to-positive numeric labeling for bipolar judgments.

Batyrshin et al. (2017) provide a comprehensive survey of bipolar measurement instruments across several fields, including psychology, sociology, medicine, recommender

systems, and sentiment analysis. It defines a bipolar rating scale as one with symmetry between negative and positive categories, where zero is a meaningful neutral point. The study discusses the structure of verbal labels, the cognitive effects of using equidistant categories, and the mathematical basis of bipolar scoring. This analysis is relevant because the effect of OTA techniques on quality attributes is similar to a semantic differential task, where the response indicates directionality (positive or negative) rather than just intensity. Unlike other references that focus on quality assessment methods, architectural tactics, or multi-criteria decision models (Christensen et al., 2010; Yılmaz and Buzluca, 2024; Moaven and Habibi, 2020; Preston and Colman, 2000; Finstad, 2010; Kazman et al., 1994; Orellana and Astudillo, 2025; Orellana et al., 2019), this article addresses the construction and validity of symmetric scales with negative and positive ranges. It also shows that common correlation measures used in non-bipolar contexts can be misleading when applied to ratings with opposing poles, emphasizing the need for scoring structures that preserve linear ordering and symmetric interpretation.

The proposal of Sangster et al. (2001) was selected because it examines how respondents interpret different numeric codings, such as traditional low-to-high positive integers and negative-to-positive integers centered on zero. This distinction is important for this study, where the polarity of the rating indicates the direction of QA impact. The authors show that negative-to-positive numbering changes how respondents perceive the scale, making it bipolar rather than unipolar. Their findings support the use of values like -2, -1, 0, $+1$, and $+2$ to reinforce a balanced continuity with a meaningful midpoint, which matches how technique impact is conceptualized in this study. By demonstrating that numeric labeling influences perceptions of symmetry and balance, this reference offers empirical support for using a negative-to-positive scale rather than a conventional Likert format with only positive integers.

The remaining references were not selected because they do not address the design, interpretation, or psychometric behavior of bipolar scales. Several focus on architectural evaluation methods, risk identification, or quality attribute interactions, such as the aSQA technique (Christensen et al., 2010), Software Architecture Analysis Method (SAAM) (Kazman et al., 1994), Architecture Tradeoff Analysis Method (ATAM) (Kazman et al., 2000), or architectural pattern trade-off techniques (Orellana et al., 2019), but do not discuss rating-scale construction or the cognitive effects of negative and positive responses. Other articles use fuzzy logic or multi-criteria decision-making, such as fuzzy logic–based maintainability evaluation (Yılmaz and Buzluca, 2024) or fuzzy Analytic Hierarchy Process (AHP) approaches (Moaven and Habibi, 2020), and do not use discrete Likert-scale judgments, so they do not inform the structure of a bipolar evaluative instrument. The study on the optimal

number of response categories (Preston and Colman, 2000) is useful for scale length but does not analyze polarity, numeric sign directionality, or the presence of a conceptual midpoint. Research comparing 5− and 7-point scales in usability contexts (Finstad, 2010) focuses on interpolation behavior and survey response accuracy, rather than bipolar semantics or negative numeric labeling.

While the alternative references (Christensen et al., 2010; Yılmaz and Buzluca, 2024; Moaven and Habibi, 2020; Preston and Colman, 2000; Finstad, 2010; Kazman et al., 1994; Orellana and Astudillo, 2025; Orellana et al., 2019) offer valuable insights for software architecture evaluation, fuzzy quality models, and QA trade-offs, they do not provide the theoretical basis needed to justify a symmetric, centered bipolar scale. The two selected articles (Batyrshin et al., 2017; Sangster et al., 2001) were chosen because they specifically address the conceptual and numerical structure required for assessing the bidirectional impact of OTA techniques on QAs. Together, they support both the validity of using a bipolar continuity and the use of negative-to-positive integer labels with a central neutral point. This makes them the most appropriate and methodologically relevant sources for designing the 5-point Likert-style bipolar scale used in this study.

Building on this foundation, the scale used in this study ranged from -2 (degradation) to $+2$ (improvement), with 0 denoting no discernible impact. Table 4.5 summarizes the interpretation of each value for QA impacts.

Table 4.5 5-point Likert-style bipolar scale to qualify each QA Impact.

Level	Description	Example
-2	Strong Negative Impact	Blockchain validation severely reduces scalability
-1	Slight Negative Impact	Encryption slightly increases energy consumption
0	Neutral (No Impact)	Metadata format doesn't affect privacy
$+1$	Slight Positive Impact	Incremental update modestly improves performance
$+2$	Strong Positive Impact	Secure boot ensures firmware reliability

This configuration is particularly well-suited to OTA techniques, whose architectural effects are rarely one-sided: a technique can strengthen one quality attribute while simultaneously stressing another. The bipolar model provides the precision and balance needed to expose these competing forces and reason cleanly about the resulting trade-offs.

4.6.2 Expert-Based Evaluation

To ensure the validity and practical relevance of the impact assignments, the assessment relied on four senior professionals with 13–17 years of experience in Software Architecture,

Table 4.6 Relevance of candidate references for the design of a symmetric bipolar rating scale

Authors	**Reason for exclusion as primary bipolar-scale reference**
Batyrshin et al. (2017)	Comprehensive survey of bipolar measurement instruments across multiple domains. Defines symmetry, neutral midpoint, and scoring models for bipolar scales; directly applicable to negative, neutral, and positive judgments.
Sangster et al. (2001)	Experimental study on how respondents interpret different numeric codings. Provides empirical evidence that negative–positive numbering changes perceived polarity and supports the use of symmetric scales.
Christensen et al. (2010)	Focuses on architectural scenario analysis and quality assessment processes. It does not address the design, polarity, or numeric structure of rating scales, and therefore cannot justify the use of a symmetric bipolar continuum.
Kazman et al. (1994)	Proposes a method for scenario-based architecture evaluation. Judgments are qualitative and method-centric; the work does not study how respondents interpret negative vs. positive categories or midpoints on a scale.
Kazman et al. (2000)	Emphasizes identification of trade-offs and risks for architectural decisions. Although it uses quality attributes, it does not discuss rating-scale semantics, bipolarity, or numeric labeling.
Orellana et al. (2019)	Addresses the analysis of architectural trade-offs and quality attribute interactions, but not the psychometric behavior of rating instruments.
Yılmaz and Buzluca (2024)	Uses fuzzy logic to evaluate quality attributes in microservice-based architectures. Evaluations are expressed via membership functions and fuzzy numbers rather than discrete bipolar categories.
Moaven and Habibi (2020)	Applies fuzzy AHP for multi-criteria decision making. It is based on pairwise comparisons and fuzzy sets, not on discrete Likert-style bipolar judgments, and thus does not support negative–positive integer labeling.
Preston and Colman (2000)	Investigates the optimal number of response categories and the impact on measurement quality. While useful for deciding on a 5-point scale, it does not analyze polarity, or sign directionality.
Finstad (2010)	Compares 5- and 7-point scales in usability evaluation, focusing on reliability and sensitivity. It assumes a unipolar, positive-only format and does not study negative-to-positive numbering or bipolar semantics.
Orellana and Astudillo (2025)	Concentrates on architectural tactics for confidentiality and security. The article provides insights into quality attributes but does not discuss rating scales or the cognitive interpretation of bipolar judgments.

IoT, Information Security, and IT Management. Their experience in designing and operating distributed systems enabled them to interpret architectural implications beyond those explicitly reported in primary studies. Table 4.7 summarizes the experts involved in the quality-attribute trade-off assessment.

Table 4.7 Experts involved in the assessment of the impact across the QAs in the techniques.

Expert	Experience (years)	Fields of Expertise
Expert 1	17	Software Architecture, IoT, Information Security
Expert 2	16	Software Architecture, Software Development
Expert 3	17	Software Architecture, IT Management, IoT
Expert 4	13	Software Architecture, Information Security

4.6.3 Evaluation Procedure

The evaluation followed a streamlined yet rigorous three-step procedure:

1. **Independent scoring**: Each expert rated all technique–QA pairs independently, ensuring unbiased reasoning and capturing variation in practitioner perspectives.

2. **Consensus session**: A focused 60-minute session was held to review divergences, clarify architectural assumptions, and converge on stable, shared interpretations of the impacts.

3. **Stability verification**: A final cross-check ensured consistency across mechanisms, eliminated asymmetries in the use of the scale, and confirmed adherence to the semantics of a strictly bipolar model.

The three steps ensured impact assessments that are internally consistent, analytically balanced, and grounded in expert architectural judgment.

4.6.4 Resulting Trade-off Model

The consolidated ratings form a comparative trade-off model that makes explicit where each OTA technique provides value, where it introduces tension, and how it affects the overall quality-attribute landscape. This model directly supports RQ1 by enabling systematic comparison across techniques and their architectural trade-offs.

Through structured evidence aggregation, multi-stage homologation, and expert-driven refinement, the methodological pipeline consolidates a heterogeneous body of OTA-related

practices into a coherent and architecturally meaningful set of techniques. Rather than being a mere byproduct of synthesis, this set represents a distilled view of recurring responsibilities, operational patterns, and architectural approaches. As a result, it provides a stable, reproducible, and conceptually grounded basis for characterizing the solution space of OTA update systems in IoT.

Chapter 5

Building DeOTA-IoT: An OTA Update Techniques Catalog for IoT

The catalog presented in this section is the consolidated output of the methodological process described previously. By integrating evidence from the literature, harmonizing terminology, and validating the resulting abstractions with experts, the study distilled a broad set of OTA practices into a structured and architecturally meaningful collection of techniques. The catalog follows the six mechanisms that define the core responsibilities of OTA update systems for IoT defined in Chapter 2, ensuring a direct alignment between architectural intent and operational realization.

The catalog is designed to serve both as a practical reference and as an analytical instrument. It supports engineers in selecting and combining OTA procedures, and it enables systematic reasoning about how these procedures affect system-level qualities in heterogeneous IoT deployments.

5.1 Techniques Characterization

Each technique is described using a uniform specification template that captures: (1) a concise title; (2) a technical description that clarifies its operational logic and architectural role; (3) representative use cases; (4) a workflow representation; and (5) an impact assessment over the selected QAs.

The techniques specified in the catalog and grouped into the six mechanisms are the following:

1. **Mechanism for secure and safe updates (M1):**

- **T.1.1 Firmware Integrity, Authenticity, and Installation Verification:** Use cryptographic hashes, such as SHA-256 or CRC (Stallings and Brown, 2017), and digital signatures, such as ECDSA or RSA (Stallings and Brown, 2017), to verify firmware integrity during transfer. Also, verification should be performed within trusted environments, and attestation mechanisms should be applied to confirm that the installation has been completed successfully. Secure boot mechanisms, such as ARM TrustZone, Secure Boot System Firmware Update (SBSFU) (Matrosov et al., 2019), or similar solutions, are used to ensure the authenticity of firmware before it is executed.
- **T1.2 Secure Payload Encryption and End-to-End Communication:** Firmware payloads should be encrypted with symmetric or asymmetric cryptographic algorithms, such as AES or RSA (Stallings and Brown, 2017), and packaged in secure formats, for example CBOR Object Signing and Encryption (COSE) (Schaad, 2022), JSON Object Signing and Encryption (JOSE) (Siriwardena, 2019), or JSON Web Signature (JWS) (Jones et al., 2015a). OTA transmissions should be protected using protocols such as Transport Layer Security (TLS), Datagram Transport Layer Security (DTLS), or Object Security for Constrained RESTful Environments (OSCORE) (Wahlström et al., 2019). The use of short-lived credentials helps to maintain confidentiality, authenticity, and resilience during communication between the Original Equipment Manufacturer (OEM) and the device.
- **T1.3 Trusted Execution and Runtime Protection:** Safeguard the integrity and confidentiality of the OTA update process during installation and execution by isolating sensitive operations within secure, trusted, or hardened execution environments (Muñoz and Fernandez, 2020; Orellana et al., 2022b,a). Secure environments include Trusted Execution Environments (TEEs), hypervisors, or hardened operating system contexts, which specifically protect cryptographic keys, update logic, and system state. By isolating these components, sensitive processes and data are shielded from compromise throughout both installation and execution phases, particularly important for devices exposed to physical access or hostile runtime conditions.
- **T1.4 Key, Credential, and Identity Management:** Generate, provision, and manage cryptographic keys and digital certificates to facilitate secure device-server authentication (Orellana et al., 2022b,a). System must maintain unique device identities, and robust authentication credentials, such as multi-factor

authentication or SIM-based methods, should be pre-installed and configured. Additionally, the System must enforce role-based access control.

- **T1.5 Decentralized and Blockchain-Based Verification:** Use blockchain, smart contracts, fog nodes, or similar decentralized methods (Choi and Lee, 2020; Witanto et al., 2020; Yohan et al., 2025; Mirsky et al., 2020) to verify firmware authenticity, track delivery, enforce constraints, and allow devices to perform transaction-based validation queries before installation.
- **T1.6 Compatibility and Threat Prevention:** Perform compatibility checks via binary analysis or simulation/digital twins before deployment (Empl et al., 2022). Detect and prevent threats using gateway-based anomaly detection, device fingerprinting, and network isolation to block unauthorized updates.
- **T1.7 Protocol, Debug Interface, and Request Security:** Prevent unauthorized update requests, protocol-level attacks, and exploitation of debug interfaces by securing communication protocols and restricting low-level system access using timestamp validation, integrating link-layer security (e.g., AUTOSAR Secure Onboard Communication (SecOC) (Soffar, 2024)), securing debug interfaces (e.g., JTAG (Parker, 2016), SWD (Yiu, 2020)), and authenticating update requests using tokens, challenge-response, or Multi-Factor Authentication (MFA) (Wilson and Hingnikar, 2019). This technique addresses threats such as replay attacks, unauthorized firmware injection, and misuse of debug ports, ensuring that OTA update requests follow authenticated and validated communication paths.
- **T1.8 Installation and Runtime Security Assurance:** Maintain long-term firmware integrity and system trust by enforcing secure installation conditions and continuous runtime verification using certificates, Root of Trust, Physically Unclonable Functions (PUFs), TEE storage, and short-lived tokens to secure installation processes and maintain firmware integrity during runtime (Regenscheid, 2018). This technique establishes a persistent root of trust that ensures only authenticated and validated firmware remains active across reboots and update cycles, supporting sustained system reliability and resistance to post-installation tampering.

2. **Mechanism for updates management (M2):**

- **T2.1 OTA Orchestration and Management Interfaces:** Use centralized or distributed OTA platforms (e.g., hawkBit (Foundation, 2025), ZT-OTA (Aoki et al., 2024)), web/mobile dashboards, and lifecycle APIs (RESTful CRUD,

polling, event triggers) (De, 2017) to manage firmware updates, deployment, and logging.

- **T2.2 Role-Based Access and Permissions:** Define update actors (e.g., OEM, integrator, uploader, user) and assign role-based permissions to control who can initiate, approve, or manage firmware updates.
- **T2.3 Workflow Coordination and Traceability:** Manage the update process through structured workflows (submission, review, authorization, deployment) with unique session identifiers for tracking and conflict prevention.
- **T2.4 Device Classification and Delivery Mode Selection:** Categorize devices by capability (secure-capable vs constrained) and choose appropriate delivery methods (WiFi direct, BLE to WiFi, BLE-only) to optimize update efficiency (Moran et al., 2021).
- **T2.5 User Awareness and Consent Management:** Notify users before updates, allow them to approve or postpone installations, and provide clear summaries of changes before and after the update.

3. **Mechanism for code dissemination, propagation, and installation (M3):**

   - **T3.1 OTA Protocols, Delivery Models, and Channel Selection:** Disseminate updates using MQTT, CoAP, HTTP(S), BLE mesh, LoRaWAN multicast, satellite, wired, or hybrid push-pull approaches (Moran et al., 2021; Bas and Dowhuszko, 2021). Select centralized, gateway, peer-to-peer, or decentralized (e.g., IPFS (Labs, 2025)) models based on scalability, reach, and device capabilities.
   - **T3.2 Manifest-Driven and Policy-Based Update Control:** Use dedicated channels for manifest distribution, including metadata, signatures, version rules, rollback constraints, and attestation data. Support deferred or event-triggered installation and coordinate updates with middleware or decentralized logic enforcing policies across devices.
   - **T3.3 Reliable Transfer and Execution Resilience:** Ensure that OTA updates are delivered and applied correctly despite network instability, device restarts, or power interruptions. This technique incorporates operations such as acknowledgments, buffering, retries, resumable transfers, and execution checkpoints to prevent partial or corrupted updates. It is especially relevant in large-scale or unreliable network environments, where resilience is required to preserve system availability and prevent inconsistent device states.

- **T3.4 Pre-Deployment Validation and Optimization:** Perform compatibility checks, sandbox testing, and module validation before deployment. Enhance targeting and performance using TinyML-based deployment (Elhanashi et al., 2024), model-specific packaging, anomaly detection, and predictive optimization.
- **T3.5 Lightweight Incremental Update Application:** Reduce update size, transmission overhead, and installation time by applying firmware or software updates incrementally rather than replacing full images. This technique updates only the modified portions of the system through insert, modify, delete, or copy operations in memory, making it particularly suitable for bandwidth-constrained, energy-limited, or intermittently connected IoT devices. Lightweight incremental updates improve availability and energy efficiency by minimizing update duration and reducing disruption during the update process.
- **T3.6 Secure Transfer, Authentication, and Installation Readiness:** Protect the OTA dissemination and installation process against unauthorized access, replay attacks, and unsafe installation conditions. Use methods such as challenge-response, One-Time Password (OTP), bound tokens, certificates, mutual authentication, or JSON Web Token (JWT) (Jones et al., 2015b). Monitor system readiness and manage update state transitions using state machines. Prevent rollback by employing write-once memory. These measures directly reduce risks such as unauthorized modification, downgrading critical software, and installing updates on unprepared devices. By combining authentication methods, update state management, and rollback prevention, the likelihood of malicious updates, software version inconsistencies, and system instability during dissemination is reduced.

4. **Mechanism for system recovery (M4):**

- **T4.1 Partition-Based Rollback and Recovery:** Use A/B partitioning, checkpointing, Non-Volatile Memory (NVM) (Hidaka, 2018), and bootloader switching to enable firmware rollback and system recovery after failed updates.
- **T4.2 Telemetry-Driven Recovery Triggers:** Monitor update outcomes via telemetry and version checks, initiating rollback if errors or inconsistencies are detected.
- **T4.3 Role-Based Recovery Initiation:** Allow authorized roles, such as firmware uploader or manufacturer, to initiate recovery using dashboards or management tools.

- **T4.4 Energy-Aware and Selective Retransmission:** Implement recovery strategies that minimize energy consumption by using selective retransmission for failed update segments.
- **T4.5 Pre-Commit Testing and Post-Recovery Validation:** Conduct update code testing before committing changes and verify system stability after recovery to ensure proper functionality.

5. **Mechanism for updates scheduling (M5):**

   - **T5.1 Context-Aware Update Scheduling:** Schedule updates based on battery thresholds, GPS or network time synchronization, retry windows, and deferred execution slots.
   - **T5.2 Event and Period-Based Update Triggers:** Initiate updates via specific events, such as repository merges, or through periodic checks of the gateway version.
   - **T5.3 Multi-Period Update Arrangement:** Organize updates into multiple periods with the ability to pause or resume specific update modules.
   - **T5.4 Role-Based Scheduling Authorization:** Enforce scheduling permissions through dashboard role assignments and require device registration/authentication before scheduling.
   - **T5.5 Policy-Driven Scheduling Enforcement:** Enforce update scheduling and execution policies based on user role, device status, or operational safety constraints.

6. **Mechanism for elaboration and packaging (M6):**

   - **T6.1 Standardized Packaging and Metadata:** Use standard update formats (SWUpdate (Babic and contributors, 2025), LwM2M (Open Mobile Alliance, 2025)) with structured metadata (JSON (T. Bray, 2017), FlatBuffers (LLC, 2025)) and delta compression for efficient update creation.
   - **T6.2 Modular Firmware Update Strategies:** Implement runtime linking, in-place updates, and dependency resolution to enable modular and flexible firmware updates.
   - **T6.3 ML Model Packaging for OTA:** Package OTA-delivered machine learning models with embedded metadata to ensure compatibility and traceability.

- **T6.4 Storage Optimization via Compression and Caching:** Use compression and flash-based caching to minimize storage usage during updates.
- **T6.5 Secure and Optimized Payload Formats:** Apply compression, delta updates, or custom payload formats with integrated versioning and encryption for secure, efficient delivery.

Each technique in the catalog is defined as an architectural decision unit, rather than as a single implementation step. In particular, techniques in the security and dissemination mechanisms may encompass multiple related practices because they address a common architectural purpose and are typically selected and evaluated together during OTA system design. For this reason, technique descriptions explicitly state not only how a solution is realized, but also why it is applied and in which contexts it is most relevant. This framing is intended to support architectural reasoning, comparison, and justification of design choices without committing to low-level implementation details at early design stages.

The implications of these characterizations are made explicit through the trade-off representation described in Section 5.3.

5.2 Quality Assesment Specification

As established in Section 4.5, the catalog uses ten quality attributes to represent how each technique affects these qualities within the architecture of an OTA update system. These attributes reflect system-level concerns that shape the design, coordination, and execution of OTA update procedures and provide a common vocabulary for characterizing the techniques across mechanisms.

Table 5.1 presents the selected attributes together with their definitions. The definitions emphasize architecturally observable properties to support consistent expert assessment and comparative reasoning across techniques. This table consolidates the terminology used throughout the catalog and clarifies the scope of each attribute before it is applied to describe the quality implications associated with the techniques.

These attributes define the quality dimensions adopted in the catalog and complete the specification of the criteria used to characterize the techniques. With these definitions in place, the foundation is set for examining how each technique influences the quality profile of an OTA update system.

The quality assessment performed in this study is intentionally grounded in software architecture quality attributes, which provide a well-established basis for reasoning about architectural trade-offs. Accordingly, the assessment focuses on how OTA techniques posi-

Table 5.1 Quality Attributes (QA) in Software Architecture and IoT, built with the QAs listed by Khezemi et al. (2024) and extended with a description for each of them.

ID	QA	Description
QA1	Security	Ability of a system to protect itself from unauthorized access, modifications, or attacks.
QA2	Scalability	Ability of a system to handle increased load or demand without degrading performance.
QA3	Performance	Ability of a system to meet the required speed, responsiveness, and resource usage.
QA4	Availability	Proportion of time that a system is functional and accessible.
QA5	Interoperability	Ability of a system to exchange data and services with heterogeneous devices, platforms, and software components using standardized interfaces, protocols, or formats, requiring minimal adaptation or integration effort.
QA6	Reliability	Ability of a system to consistently perform its required functions under specified conditions.
QA7	Privacy	Ability to safeguard personal and sensitive information collected by devices, ensuring it is only used for its intended purpose.
QA8	Energy Management	Ability to use energy efficiently across devices, ensuring optimal performance while minimizing power consumption.
QA9	Flexibility	Ability of a system to adapt to future requirements or changes in environment with minimal disruption.
QA10	Evolvability	Ability of a system to accommodate functional, technological, or operational changes over time with limited architectural restructuring, low implementation effort, and controlled impact on existing system behavior.

tively or negatively influence attributes such as security, scalability, performance, availability, interoperability, reliability, privacy, energy management, flexibility, and evolvability.

Economic cost and implementation time are not treated as quality attributes within this specification. While these factors are undeniably important in engineering practice, they are highly context-dependent and vary according to organizational structures, existing infrastructure, development processes, tooling ecosystems, and team expertise. As a result,

they cannot be consistently or objectively characterized at the architectural technique level across heterogeneous IoT contexts. For these reasons, economic and implementation cost modeling is considered outside the scope of the present quality-oriented assessment and is identified as an important direction for future work, complementing the architectural insights provided by the DeOTA-IoT catalog.

5.3 Characterization and Representation of Trade-offs

The impact assignments derived from the expert evaluation are integrated into the catalog to make the architectural consequences of each technique explicit. These values indicate how every technique influences each quality attribute and form a comparable impact profile across mechanisms.

Table 5.2 compiles these ratings in a unified structure. Each cell reflects the expert-validated effect of a technique on a specific attribute, enabling direct comparison and exposing the contrasting quality tendencies that characterize OTA update solutions. By presenting the impacts in this consolidated form, the catalog reveals the trade-offs inherent to OTA design and provides a clear basis for selecting and combining techniques according to system-level objectives.

The trade-off characterization presented in this subsection operationalizes the DeOTA-IoT catalog as a decision-support instrument rather than a descriptive listing of techniques. During OTA system design, architects typically select candidate techniques within each mechanism based on functional requirements and deployment constraints. The trade-off profiles provided in Table 5.2 enable designers to explicitly evaluate how each candidate technique strengthens or stresses relevant quality attributes, such as security, availability, energy management, or evolvability.

In the overall procedure, trade-off characterization serves three complementary purposes. First, it supports intra-mechanism comparison, allowing architects to contrast alternative techniques that fulfill the same architectural responsibility but exhibit different quality impacts. Second, it enables cross-mechanism reasoning, making visible how the combined selection of techniques may introduce reinforcing or conflicting quality effects across the OTA workflow. Third, it provides a transparent rationale for architectural decisions, supporting justification, communication, and later reassessment as system requirements evolve.

To support transparency and reuse of the proposed artifact, the DeOTA-IoT techniques catalog has been made publicly available through the Zenodo research data repository under DOI: https://doi.org/10.5281/zenodo.17404292.

Table 5.2 Techniques and their corresponding QAs' impact, organized by mechanisms and specified with each technique ID.

Mechanism	Technique	QA1	QA2	QA3	QA4	QA5	QA6	QA7	QA8	QA9	QA10
M1	T1.1	+2	+1	0	+1	+1	+2	0	−1	−1	0
	T1.2	+2	+1	−1	+1	+1	+1	+2	−1	+1	0
	T1.3	+2	−1	0	+1	−1	+2	+1	−1	−1	+1
	T1.4	+2	+1	−1	+1	+2	+2	+2	−1	+1	+1
	T1.5	+2	−2	−1	+1	−1	+1	−1	−2	+1	+1
	T1.6	+2	+1	−1	+1	+1	+2	0	−1	+1	+1
	T1.7	+2	+1	0	+1	+1	+2	+1	−1	+1	+1
	T1.8	+2	+1	0	+2	−1	+2	+1	−1	−1	+1
M2	T2.1	+1	+2	+1	+1	+2	+1	0	0	+1	+1
	T2.2	+2	+1	0	+1	+1	+1	+1	0	+1	+1
	T2.3	+1	+1	0	+1	+1	+2	0	0	+1	+1
	T2.4	+1	+2	+1	+1	+1	+1	0	+2	+2	+1
	T2.5	+1	0	0	+1	0	+1	+2	0	+1	+1
M3	T3.1	+1	+2	+1	+1	+2	+1	0	+1	+2	+1
	T3.2	+2	+1	+1	+1	+1	+2	0	0	+1	+2
	T3.3	+1	+1	+1	+2	0	+2	0	−1	0	+1
	T3.4	+1	+1	+1	+1	+1	+2	0	−1	+1	+1
	T3.5	0	+1	+2	+1	0	+1	0	+2	+1	+1
	T3.6	+2	+1	+1	+1	+1	+1	+1	−1	+1	+1
M4	T4.1	+1	+1	+1	+2	0	+2	0	−1	+1	+1
	T4.2	+1	+1	0	+2	0	+2	0	−1	+1	+1
	T4.3	+2	+1	0	+1	0	+1	+1	0	+1	+1
	T4.4	0	+1	+1	+1	0	+1	0	+2	+1	+1
	T4.5	+1	+1	0	+2	0	+2	0	−1	+1	+1
M5	T5.1	0	+1	+1	+2	0	+2	0	+2	+1	+1
	T5.2	+1	+1	0	+1	0	+1	0	0	+1	+1
	T5.3	0	+1	+1	+2	0	+1	0	+1	+2	+1
	T5.4	+2	+1	0	+1	0	+1	+1	0	+1	+1
	T5.5	+2	+1	+1	+1	0	+2	0	0	+1	+2
M6	T6.1	+1	+1	+1	0	+2	+1	0	0	+1	+1
	T6.2	+1	+1	+1	+1	0	+1	0	+1	+2	+2
	T6.3	+1	+1	+1	+1	+1	+1	0	0	+1	+2
	T6.4	0	+1	+2	+1	0	+1	0	+2	0	+1
	T6.5	+2	+1	−1	+1	+1	+1	0	+1	+1	+1

Legend: *QA1: Security, QA2: Scalability, QA3: Performance, QA4: Availability, QA5: Interoperability, QA6: Reliability, QA7: Privacy, QA8: Energy Management, QA9: Flexibility, QA10: Evolvability, M1: Mechanism for secure and safe updates, M2: Mechanism for updates management, M3: Mechanism for code dissemination, propagation, and installation, M4: Mechanism for system recovery, M5: Mechanism for updates scheduling, M6: Mechanism for elaboration and packaging.*

Chapter 6

Experimental Validation

To assess the practical value of the catalog as a design support instrument for OTA update systems, a controlled experiment in an industrial setting was conducted. The study examined whether the catalog enables practitioners to produce OTA architectures that are more complete, accurate, and internally consistent than those derived solely from professional expertise.

Industrial experiments commonly operate with reduced groups due to availability and operational constraints in companies (Felderer and Travassos, 2020). In such settings, small sample sizes are standard, and robust effect–size estimators provide stable insights despite small n (Kitchenham and Madeyski, 2024). For this reason, although the experiment involved only ten participants, the analysis does not rely on statistical significance testing, whose power would be insufficient for the sample size. Instead, the evaluation centers on performance metrics and nonparametric effect–size estimators such as the Vargha–Delaney A_{12} (Vargha and Delaney, 2000) and Cliff's δ (Cliff, 1993). These estimators provide distribution-free assessments of practical differences between conditions and are well-suited to small-sample experimental designs. Their use allows the study to quantify the magnitude and direction of the observed effects and to assess whether the empirical patterns reflect practically meaningful differences between the evaluated conditions.

6.1 Experimental Design

The experimental design establishes the controlled setting for evaluating the catalog's contribution to OTA architectural decision-making. It defines the hypotheses, participant groups, tasks, materials, and evaluation criteria, following established empirical software engineering guidelines (Wohlin et al., 2012; Felderer and Travassos, 2020).

6.1.1 Objectives and Hypotheses

The objective of the experiment was to determine whether the catalog measurably improves architectural decision-making in OTA update systems.

The hypotheses guiding the experiment are:

- H_0: The use of the catalog does not have a significant impact on the quality or efficiency of OTA system design.
- H_1: The use of the catalog significantly enhances the quality and efficiency of OTA system design, leading to more systematic decisions that align with key quality attributes such as security, maintainability, and interoperability.

6.1.2 Context and Participants

The experiment was conducted at Coffee Electronics Pte. Ltd., a multinational company specializing in embedded and IoT engineering, at its main operational center in Colombia. All members of the technology division were invited to participate, and the ten professionals who volunteered met the inclusion criteria of prior experience in distributed systems or IoT development. All participants met the inclusion criteria of having prior professional experience with distributed systems or IoT development, and no participants were excluded after enrollment.

The participant pool reflects the typical composition of an industrial IoT engineering team. Their backgrounds spanned software development, firmware development, embedded systems, and cloud–IoT integration, with experience levels ranging from early-career practitioners to senior engineers with more than a decade of industry experience. Table 6.1 summarizes the participants, their experience, and the group labels used later in the experimental setup.

6.1.3 Experimental Setup

Participants were allocated to two experimental conditions using a stratified randomization procedure (Wohlin et al., 2012) based on seniority. This approach balanced junior and senior practitioners across groups, reducing the likelihood that experience-related differences would influence the outcomes.

The two groups were defined as follows:

- **Control group (G_1):** Completed the OTA design task without access to the catalog and relied exclusively on their prior knowledge and professional experience.

Table 6.1 Participants, experience, and technical background.

Group ID	Participant ID	Experience (years)	Technical Domain
G_1	P1	13	Web and Mobile Development
	P2	2	Software Development
	P3	1	Software Development
	P4	1	Firmware Development
	P5	5	Firmware Development
G_2	P6	14	Hardware
	P7	14	Software, Telematics and Cloud
	P8	1	Hardware
	P9	2	FPGA, DSP, and Embedded Systems
	P10	2	FPGA

Legend: *FPGA: Field-Programmable Gate Array, DSP: Digital Signal Processor.*

- **Experimental group (G_2):** Completed the same task with access to the catalog and received a short orientation on its structure and intended use.

Both groups addressed the same OTA design scenario, which involved heterogeneous devices, intermittent connectivity, confidentiality and integrity constraints, authenticity verification, and rollback protection. The task required participants to analyze this scenario and identify the architectural techniques they considered appropriate for addressing it.

The experimental session lasted 110 minutes and followed a structured sequence of activities to ensure consistent conditions across groups. This duration was deliberately selected to support reflective architectural reasoning while remaining feasible within an industrial setting. A session length of this magnitude is consistent with established practices in controlled industrial experiments in software engineering (Osses et al., 2018; Orellana and Astudillo, 2025; Orellana et al., 2024), where durations of 90-120 minutes are commonly used to balance task complexity, cognitive load, and participant fatigue. By regulating the flow of information, standardizing materials, and allocating dedicated time for independent reasoning, the session design ensured that the collected data reflected deliberate architectural judgment rather than procedural artifacts.

During the session, all participants worked individually under controlled conditions, using identical materials and tools. The session was monitored to ensure procedural consistency, mitigate bias, and maintain data reliability throughout all phases of the experiment.

Table 6.2 summarizes the procedure, including the content and duration of each phase of the session.

Table 6.2 Structure and timing of the experimental session.

Activity	Description	Duration
1. Introduction & Training	Presentation of the experiment's objectives, group allocation, and a short training on the concept of architectural techniques to ensure a shared understanding of terminology.	15 min
2. Materials Delivery	Distribution of the complete materials package: OTA design scenario, detailed written instructions, and standardized response templates to both groups. The experimental group additionally received the catalog.	10 min
3. Catalog Briefing (G_2 only)	Orientation on the catalog's structure, contents, and examples of technique selection for the experimental group.	10 min
4. Design Activity	Independent design of the OTA subsystem. The experimental group (G_2) applied the catalog during the task, while the control group (G_1) relied solely on professional experience.	60 min
5. Debriefing & Submission	Submission of the selected catalog techniques for the OTA scenario, followed by a brief feedback session.	15 min

All materials used in the experiment, including the scenario description, templates, and supporting documentation, are publicly available at Zenodo (https://doi.org/10.5281/zenodo.17404292), ensuring full reproducibility.

6.1.4 Scenario: LaboTech Remote Lab System

This subsection presents the scenario used as the foundation for the experimental validation. It defines the system, its operational environment, and the conditions that motivate its update needs, providing the realistic context against which architectural decisions and OTA technique selections were evaluated.

a) System Description The system requiring an OTA update mechanism consists of multiple remote laboratory stations deployed across teaching facilities. Each station operates as a host PC, accessible remotely by students and instructors, and is responsible for controlling a heterogeneous set of laboratory instruments. Typical equipment connected to each station includes digital oscilloscopes, programmable power supplies, arbitrary waveform generators, cameras, and auxiliary electronic boards, interfaced through USB, LAN, and UART connections.

Each laboratory station runs a full operating system and integrates multiple software layers, including operating system packages, device drivers, manufacturer-provided SDKs, firmware update utilities for connected instruments, and experiment-control scripts developed in languages such as Python and Bash. As a result, the OTA update system must support updates to operating system components, application software, drivers, scripts, and firmware for external instruments.

The number of stations and the specific mix of instruments may vary across laboratories, reflecting realistic heterogeneity in hardware revisions, firmware versions, and connectivity constraints. This variability was intentionally preserved in the scenario to evaluate architectural decision-making in OTA under conditions representative of real industrial and academic remote laboratory environments.

The scenario represents a class of remote laboratory deployments rather than a single fixed installation.

b) Users and Usage Students, professors, and technical assistants reserve time slots between 08:00 h and 22:00 h. The stations usually remain inactive during the night, although during exam periods, some classes extend late into the evening. During the sessions, data and video are transmitted in real time, and the results are stored in a shared network folder.

In the experiment, participants used this usage context to design an OTA update solution at the architectural level by selecting and justifying mechanisms and techniques, rather than implementing or configuring a concrete system.

c) System Requirements The following are the main properties that the system must exhibit:

- **Performance:** The OTA system must maintain optimal performance by executing update-related activities without degrading the normal operation of the laboratory stations, particularly during active user sessions. This includes avoiding noticeable delays, excessive CPU or memory consumption, network congestion, or interference with real-time data acquisition and video streaming. Update mechanisms should therefore be designed to operate in the background, leverage incremental or staged updates, and respect scheduling constraints to ensure that ongoing experiments and remote interactions remain responsive and uninterrupted.

- **Availability:** The OTA update infrastructure must ensure that update packages, repositories, and update servers remain accessible when required during the update process. This includes maintaining the availability of distribution endpoints to prevent update

failures caused by server downtime, transient connectivity issues, or incomplete package retrieval. The OTA process should support retry, resume, or fallback mechanisms to tolerate the temporary unavailability of update services without compromising the consistency of deployed devices.

- **Interoperability:** The OTA system must support updates across heterogeneous platforms, operating systems, applications, and subsystems using standardized or widely adopted protocols and interfaces. This includes accommodating diverse hardware architectures, communication stacks, and software environments while minimizing the need for platform-specific customization, thereby enabling consistent update management across all laboratory stations.
- **Security:** The OTA system must ensure that updates are applied using secure mechanisms that protect against unauthorized access, tampering, and malicious attacks affecting devices, software components, and communication channels. This includes enforcing authentication and authorization of update sources, verifying the integrity and authenticity of update artifacts, and safeguarding update processes against replay, rollback, or man-in-the-middle attacks to preserve system trustworthiness.

d) Factors Driving Frequent Updates

- Changes in the semester curriculum (new experiment scripts, new driver versions, and the addition of instrument models).
- Application of security patches for the operating system, runtime environments, and remote access components.
- Firmware updates for instruments are provided by manufacturers (installed using the host utilities).
- The university requires updates to security certificates and the VPN client for remote access.

e) Updatable Artifacts

- **Host operating system packages**: These include the Linux kernel or cumulative operating system updates.
- **Runtime environments and libraries**: Python environments, software packages, Java libraries, and associated scripts.

- **Experimental software**: Applications, code repositories, configuration files, and experiment templates.
- **Instrument firmware**: Updated using manufacturer tools over USB or LAN; sometimes requires exclusive access and device reboot.
- **Drivers**: USB adapters, camera capture drivers, and kernel modules.
- **Configurations and policies**: Device whitelists, instrument assignment settings, reservation schedules, logging levels, and VPN profiles.
- **Licenses**: License files and firmware for USB keys (dongles) used with specific SDKs.

f) Environment Constraints

- **Topology**: USB hubs connected in chains; occasionally, faulty cables; users may turn instruments off.
- **Heterogeneity**: Multiple instrument revisions across rooms; the exact model may have different baseline firmware versions.
- **Network**: Use of campus restrictions; some laboratories are in segmented VLANs; limited nighttime bandwidth; some buildings require a proxy.
- **Storage**: Limited free space due to raw data dumps.
- **Remote access**: Remote access tools must remain available; students should not see installer pop-up windows.

g) Operational Rules

- **No interruptions during reservations**: Updates can only be performed during maintenance windows or when the station is inactive.
- **Safety**: During any instrument firmware update, outputs must remain de-energized (power-supply output off, loads disabled).
- **Atomicity**: Experimental scripts and their associated runtime versions must be updated together.
- **Compatibility control**: Firmware/driver combinations for instruments are subject to a model- and revision-specific compatibility matrix.

- **Human factors**: Laboratory staff may postpone a station update by up to 24 hours with short notice (due to exams or demonstrations).

The scenario intentionally abstracts away exact device counts and hardware models to focus the experimental task on architectural reasoning about OTA mechanisms and techniques rather than on platform-specific implementation details.

6.1.5 Ground Truth Construction and Comparison Model

To ensure methodological rigor and analytical consistency, a Ground Truth (GT) was established as the canonical reference model for this experiment. The GT encapsulates the set of techniques that represent an expert-level solution for the OTA update scenario in IoT systems.

The complete universe of techniques available for consideration is denoted as U. This universe comprises the full catalog of OTA-related architectural techniques, which collectively span reliability, confidentiality, integrity, maintainability, and scalability concerns:

$$U = \{\,\mathrm{T1.1}, \mathrm{T1.2}, \ldots, \mathrm{T1.8}, \mathrm{T2.1}, \ldots, \mathrm{T6.5}\,\} \tag{6.1}$$

To construct the GT, the expert panel followed a structured, multi-stage consensus protocol designed to ensure rigor, transparency, and stability in the resulting configuration. Each expert first reviewed all techniques in the universe U independently, assessing their necessity, relevance, and contribution to the quality attributes required by the scenario. These individual assessments were then compared to identify divergences and areas of disagreement. Through facilitated discussion, the panel examined the assumptions behind each position, considered scenario-specific constraints, and trade-offs. Using evidence-based argumentation and iterative refinement, the experts resolved inconsistencies and converged on a minimal but sufficient set of techniques deemed indispensable for achieving a secure and dependable OTA update system.

This consensus protocol was operationalized through a sequence of workshops involving the experts listed in Table 4.7. During these sessions, each candidate technique in U was analyzed in terms of its contextual relevance, architectural implications, and contribution to the target quality attributes. The process continued until unanimous agreement was reached on the final configuration representing the canonical solution:

$$\begin{aligned} GT = \{\,&T1.1, T1.2, T1.3, T1.4, T1.6, T1.7, T1.8, T2.1, T2.4, T3.1, T3.3, T3.4, \\ &T3.5, T3.6, T4.1, T4.3, T4.5, T5.5, T6.2, T6.3, T6.4\,\} \end{aligned} \tag{6.2}$$

Since the control group did not have access to the catalog during the experimental task, a subsequent mapping phase was supported by the same expert panel to ensure methodological equivalence. In this process, the experts analyzed the control group's responses, interpreting each expressed concept and associating it with the technique from the catalog that was most semantically aligned. This mapping ensured that both groups could be represented within the same formal domain U, thus eliminating ambiguity and enabling valid comparative analysis.

Each participant p generated a subset of techniques reflecting the choices made during the design session. This set was denoted as S_p, where $S_p \subseteq U$, indicating that S_p contains exactly the techniques from the universe U that were selected by participant p. Because each participant reasons independently, the resulting subsets differ across participants even though the design space remains shared.

The relationship between each participant's subset S_p and the GT provides the basis for quantifying how closely individual design decisions align with expert architectural rationale. To operationalize this comparison, participant selections were characterized using the standard four categories of set-theoretic alignment: true positives, false positives, false negatives, and true negatives. These categories are defined formally as follows:

$$TP_p = |S_p \cap GT|, \tag{6.3}$$

$$FP_p = |S_p \setminus GT|, \tag{6.4}$$

$$FN_p = |GT \setminus S_p|, \tag{6.5}$$

$$TN_p = |U \setminus (GT \cup S_p)| \tag{6.6}$$

Formally, true positives (TP) denotes techniques correctly aligned with the GT, false positives (FP) those incorrectly introduced, false negatives (FN) the relevant techniques omitted, and true negatives (TN) the non-essential ones correctly excluded. The variables TP, FP, FN, and TN form the analytical basis for the performance metrics discussed in the next section.

6.1.6 Performance Metrics

To assess the extent to which the proposed catalog influences the quality of architectural decisions in OTA system design, this study employs four performance metrics used in prior empirical evaluations of architectural decision-making: Precision, Recall, F1, and Accuracy (Osses et al., 2018; Orellana et al., 2024; Orellana and Astudillo, 2025). These metrics

provide complementary views on participants' ability to select techniques that align with the predefined GT.

Although the experiment is exploratory and does not involve statistical hypothesis testing due to its small sample size, these quantitative indicators offer a robust basis for examining whether the evidence supports or contradicts the defined hypotheses. Accordingly, they are appropriate for evaluating the practical influence of the catalog in this controlled setting. Table 6.3 summarizes their definitions and the notation used throughout the analysis.

Table 6.3 Metrics utilized in analyzing the experimental validation of the techniques catalog.

Metric	Description	Formula	
Precision	The ratio of true-positive selections to the total number of techniques chosen by a participant, indicating the correctness of their selections.	$\text{Precision} = \frac{TP}{TP+FP}$	(6.7)
Recall	The proportion of techniques present in the GT that were correctly selected.	$\text{Recall} = \frac{TP}{TP+FN}$	(6.8)
F1	A harmonic-mean aggregation of Precision and Recall that captures their joint performance in a single, balanced indicator.	$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$	(6.9)
Accuracy	The proportion of all evaluated techniques for which the participant made the correct decision, combining both true positives and true negatives.	$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$	(6.10)

Legend: *TP: True Positives, TN: True Negatives, FP: False Positives, FN: False Negatives.*

At the analysis stage, the values of Precision, Recall, F1, and Accuracy will be summarized using descriptive statistics (mean, median, and standard deviation) and compared across groups using non-parametric effect-size measures. In particular, the Vargha–Delaney A_{12} statistic (Vargha and Delaney, 2000) and Cliff's δ (Cliff, 1993) will be used to quantify the magnitude of the differences between G_1 and G_2. These measures are appropriate for

small-sample, non-normal datasets and provide a robust indication of practical differences between groups without requiring formal hypothesis testing (Kitchenham and Madeyski, 2024).

6.2 Experimental Results

This section presents an empirical analysis of how participants approached the OTA design task and how their decisions converged toward the established ground truth. The experimental results are based on the techniques selected by each participant for the given scenario, together with the written justifications they provided to explain their architectural decisions and quality-attribute considerations. Participants submitted these artifacts electronically after completing the session, using a common submission channel to ensure consistency across both experimental groups.

Using the previously defined performance metrics, the structure, consistency, and adequacy of the selected techniques are assessed, with particular attention to how participants balanced completeness and selectivity. The analysis characterizes decision-making behavior under each condition and highlights the effect of structured design support on architectural reasoning.

Table 6.4 reports the individual outcomes of the experimental task, detailing the TP, TN, FP and FN counts for each participant and the corresponding performance metrics. Figure 6.1 summarizes the Precision, Recall, F1, and Accuracy values obtained from the experimental validation across study participants.

Table 6.4 Results of the experimental validation, organized by metrics and study participants.

	G_1					G_2				
Metric	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
TP	1	3	4	2	5	13	10	8	16	8
TN	11	11	10	10	11	10	10	11	4	13
FP	2	2	3	3	2	3	3	2	9	0
FN	20	18	17	19	16	8	11	13	5	13
Precision	0.33	0.60	0.57	0.40	0.71	0.81	0.77	0.80	0.64	1.00
Recall	0.05	0.14	0.19	0.10	0.24	0.62	0.48	0.38	0.76	0.38
F1	0.09	0.23	0.29	0.16	0.36	0.70	0.59	0.52	0.69	0.55
Accuracy	0.35	0.41	0.41	0.35	0.47	0.68	0.59	0.56	0.59	0.62

As seen in Table 6.4, participants in G_1 exhibit a consistent pattern: low Recall, low F1, and high false-negative counts. Although Precision is reasonable in some cases, the collective behavior indicates that these practitioners identified only a small portion of the techniques required for an OTA update design. This pattern is entirely expected: without access to the catalog, participants must rely solely on their individual experience, which naturally limits the range of techniques they consider. As a result, their selections show a reduced exploration of the solution space and a tendency to identify only a subset of the techniques relevant to the OTA update system.

On the other hand, participants in G_2 display a distinctly different behavior. Their Recall is substantially higher, false negatives decrease accordingly, and F1 improves coherently. Accuracy also increases, suggesting that correct selections drive the improvement. These results indicate that enabled participants to conduct a more deliberate examination of the relevant techniques, organizing their decisions around the architectural concerns central to OTA updating rather than ad-hoc reasoning.

The contrast between the two conditions shows that the catalog functions as an effective design aid: it broadens the portion of the architectural solution space that participants consider and improves the alignment of their decisions with the GT. The shift is not only quantitative but conceptual, revealing a move from intuition-led choices toward more structured and principled architectural reasoning.

While Table 6.4 reports the complete set of evaluation outcomes, including TP, TN, FP, and FN counts, Figure 6.1 provides a complementary visual summary of the derived performance metrics. The column graph highlights the distribution of Precision, Recall, F1, and Accuracy values across individual participants, enabling a direct visual comparison between experimental conditions.

Figure 6.1 reveals several patterns that are not immediately apparent from the tabular results alone. Most notably, the performance profiles of G_1 and G_2 are visually well separated, indicating qualitatively different reasoning behaviors rather than incremental performance differences. Whereas G_1 exhibits irregular and heterogeneous metric profiles across participants, G_2 shows more coherent and aligned patterns, particularly for Recall and F1, suggesting increased consistency in architectural decision-making.

The figure also highlights a shift in the relationship between Precision and Recall. In G_1, moderate Precision coexists with very low Recall, reflecting selective but incomplete reasoning. In contrast, G_2 achieves higher Recall without a systematic loss of Precision, indicating that the broader coverage is driven by informed selection rather than indiscriminate inclusion. The concurrent improvement and co-movement of Recall, F1, and Accuracy across

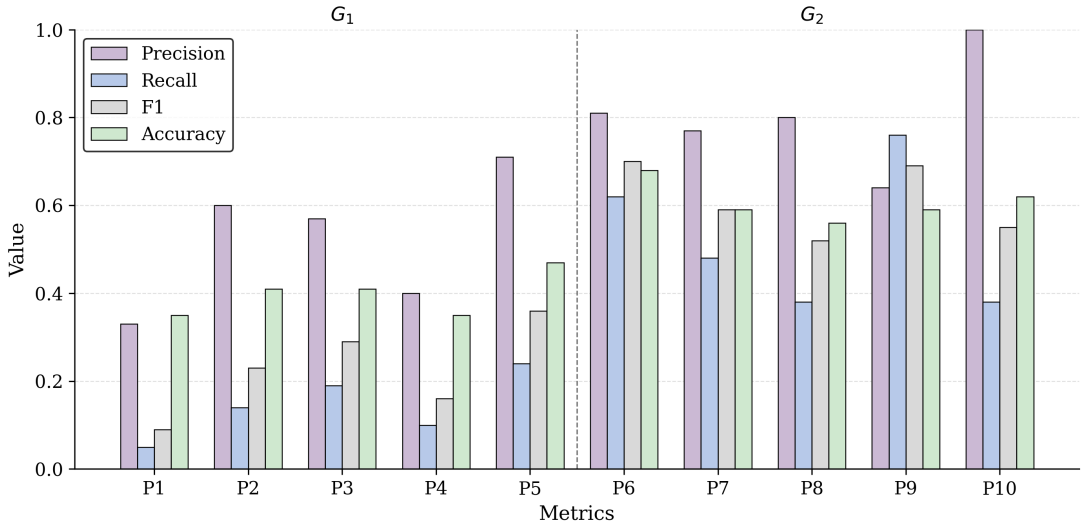


Fig. 6.1 Precision, Recall, F1, and Accuracy values obtained from the experimental validation, organized by metric and study participant.

G_2 participants further suggests that the catalog reshapes the decision process itself, leading to more systematic and stable architectural reasoning.

6.2.1 Statistical Analysis

Building on the individual-level analysis, an assessment is conducted to determine whether the observed decision-making patterns persist at the group level. Table 6.5 reports the aggregated performance metrics for G_1 and G_2, including measures of central tendency and dispersion. This aggregation reveals whether per-participant differences represent isolated cases or stable, condition-specific behaviors. By examining the mean, median, and standard deviation of Precision, Recall, F1, and Accuracy, the dominant reasoning tendencies of each group are characterized, and the catalog's influence on the overall quality and coherence of the architectural solutions is assessed.

The aggregated statistics in Table 6.5 demonstrate that the two groups follow markedly different decision patterns. G_1 exhibits consistently low Recall, with both the mean and median at 0.14 and a slight standard deviation of 0.07. This combination indicates a uniform tendency to omit essential techniques, confirming that the behavior observed at the individual level is systematic rather than participant-specific. Precision in G_1 is moderate, but the pronounced gap between Precision and Recall shows that participants made few selections and rarely identified the techniques required by the GT.

Table 6.5 Descriptive comparison of evaluation metrics between G_1 and G_2.

Metric	Statistic	G_1	G_2
Precision	Mean	0.52	0.80
	Median	0.57	0.80
	Std. Dev	0.15	0.13
Recall	Mean	0.14	0.52
	Median	0.14	0.48
	Std. Dev	0.07	0.16
F1	Mean	0.23	0.61
	Median	0.23	0.59
	Std. Dev	0.11	0.08
Accuracy	Mean	0.40	0.61
	Median	0.41	0.59
	Std. Dev	0.05	0.05

In contrast, G_2 shows substantially higher Recall (mean 0.52) with increased standard deviation (0.16), a pattern that aligns with the reduction in false negatives observed earlier. This suggests that the catalog enabled a broader and more effective exploration of the design space. At the same time, Precision in G_2 remains high (mean 0.80) with slightly lower dispersion than G_1, indicating that improved coverage did not come at the cost of increased false positives. These trends combine to produce a higher and more stable F1 score in G_2, reflecting a balanced improvement in completeness and correctness.

Accuracy follows the same direction, rising from a mean of 0.40 in G_1 to 0.61 in G_2, with similar standard deviation, indicating that the improvement is consistent across participants. Altogether, the statistical profiles indicate that the catalog supports more comprehensive and coherent architectural decision-making, increasing the recovery of relevant techniques while maintaining selection quality and reducing divergence in participant performance.

To complement the aggregated statistics reported in Table 6.5, Figure 6.2 synthesizes the average values of all evaluation metrics and places them in a directly comparable form. This broader perspective allows to appreciate how the metrics relate to one another and how consistently they shift across groups. By examining the mean levels of Precision, Recall, F1, and Accuracy side by side, the figure clarifies the overall performance profile produced by the catalog and highlights patterns that are less evident when considering the metrics individually.

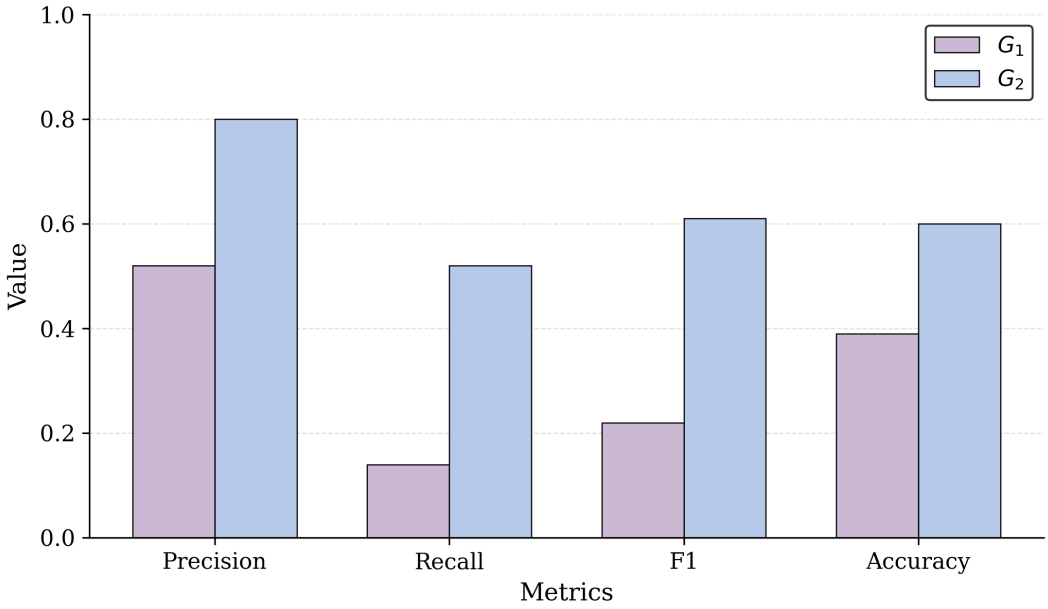


Fig. 6.2 Average values for Precision, Recall, F1, and Accuracy, illustrating the comparative effectiveness across the assessed configurations.

Figure 6.2 shows a clear and consistent performance gap favoring G_2 across all evaluated metrics. The most pronounced difference appears in Recall, where G_2 attains values several times higher than G_1, underscoring the systematic difficulty that unaided participants faced in identifying a sufficient portion of the techniques required for the scenario. The catalog enabled much broader, more complete coverage of the GT, confirming that its impact extends beyond incremental gains.

Precision also increases substantially for G_2, which indicates that the expanded coverage did not come at the cost of selecting irrelevant techniques. This balance between higher Recall and maintained Precision leads to a pronounced improvement in F1, reflecting more coherent, complete, and structurally aligned architectural solutions. Accuracy shows a similar upward shift, suggesting that the improvements in true positive identification extend to overall classification of relevant and irrelevant techniques, rather than being confined to specific aspects of the task.

A particularly relevant aspect of these aggregated comparisons is that the improvements observed in G_2 follow a coherent structural pattern rather than appearing as isolated advantages in specific metrics. The parallel increases in Recall, Precision, F1, and Accuracy suggest that the catalog influenced how participants conceptualized the task, guiding them

toward a more systematic understanding of which techniques were appropriate for the scenario. This alignment across metrics indicates that the catalog did not merely increase the quantity of techniques selected but also improved the internal consistency of those selections, revealing a qualitative shift in how participants constructed their solutions.

These results outline a coherent advantage for G_2 and make evident that the improvements observed across metrics are neither marginal nor confined to isolated aspects of the task. Rather, they point to a marked shift in how participants approached identifying relevant techniques.

Although the previous analyses clarified both individual outcomes and aggregated differences between groups, they do not reveal how evenly these improvements are distributed across participants. Understanding whether the catalog produced a broadly shared shift or whether some participants benefited more than others requires examining the internal distribution of the results. Figure 6.3 provides this view by displaying the spread and central tendency of each metric, offering insight into the consistency and variability of performance within each group.

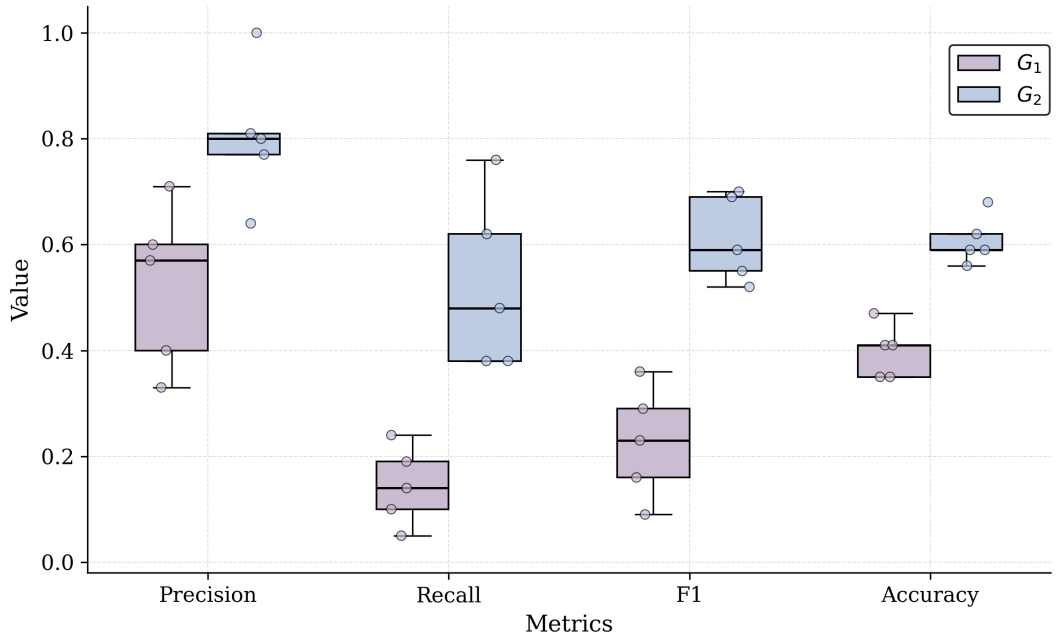


Fig. 6.3 Representation of the distribution values across Precision, Recall, F1, and Accuracy, illustrating the tendency and dispersion observed during the experimental assessment.

As shown in Figure 6.3, the distributions reveal clear and systematic differences between the two groups. In G_1, the Recall and F1 values are concentrated near the lower end of the scale, with tight interquartile ranges and no high observations. This pattern indicates that all participants performed at similarly limited levels, consistent with the unaided condition. Without access to the catalog, the range of techniques considered remains narrow, and the resulting solutions reflect uniformly low completeness.

The distributions in G_2 present a markedly different profile. Recall rises substantially and spans a broader vertical range, indicating that several participants identified a considerably larger subset of relevant techniques once the catalog was available. Precision is higher and more tightly grouped, suggesting a stable ability to identify relevant techniques across participants. The F1 and Accuracy distributions shift upward as well, with boxes positioned clearly above those of G_1 and only moderate variability, which reflects more balanced and coherent selections in the supported condition.

The boxplots, therefore, provide a distributional view that strengthens the previously observed contrasts. G_1 remains clustered around low completeness and restricted identification, whereas G_2 shows consistently higher correctness and broader coverage of the relevant techniques for the scenario. The shape and position of the boxes across metrics indicate that the improvements are systematic rather than isolated, highlighting a meaningful shift in how participants approached the design task when supported by the catalog.

6.2.2 Effect Size Analysis

Beyond examining differences in central tendency and dispersion, it is essential to quantify the magnitude of the observed improvements to determine whether the catalog produces effects of practical significance. Effect sizes characterize the strength of the contrast between G_1 and G_2 independently of sample size, making them suitable for small-sample experimental designs (Kitchenham and Madeyski, 2024). To this end, two non-parametric effect size measures that are standard in empirical software engineering were computed: the Vargha–Delaney A_{12} statistic (Vargha and Delaney, 2000) and Cliff's δ (Cliff, 1993).

For A_{12}, the guidelines of Vargha and Delaney (2000) were adopted, where values near 0.5 indicate negligible dominance and thresholds of 0.56, 0.64, and 0.71 denote small, medium, and large effects, respectively. Cliff's δ followed the magnitude thresholds introduced by Romano et al. (2006), which classify effects as negligible ($|\delta| < 0.147$), small ($|\delta| < 0.33$), medium ($|\delta| < 0.474$), and large ($|\delta| \geq 0.474$).

These measures capture stochastic dominance between groups and provide interpretable thresholds for judging practical importance. Table 6.6 summarizes the results for all performance metrics.

Table 6.6 Effect size results for the comparison between G_1 and G_2 using Vargha–Delaney A_{12} and Cliff's δ, including separate magnitude interpretations.

Metric	$\mathbf{A_{12}}$	**Cliff's δ**	**Effect Size ($\mathbf{A_{12}}$)**	**Effect Size (δ)**
Precision	0.96	0.92	Large	Large
Recall	1.00	1.00	Large	Large
F1	1.00	1.00	Large	Large
Accuracy	1.00	1.00	Large	Large

The effect size results in Table 6.6 reveal an exceptionally strong and consistent dominance of the G_2 across all evaluation metrics. The values of A_{12} and Cliff's δ fall at the upper end of their respective scales, with every metric classified as a large effect according to established interpretation criteria. This indicates that the improvements observed are not only statistically meaningful but also of substantial practical relevance in the context of architectural decision-making.

The magnitude of these effects follows directly from the structure of the data. In all metrics, every observation in G_2 exceeds every corresponding observation in G_1, yielding complete stochastic dominance of the experimental group. Under these conditions, the Vargha–Delaney statistic necessarily attains its upper bound of $A_{12} = 1.00$, reflecting that a randomly selected participant from G_2 will outperform one from G_1 with probability one. Cliff's δ mirrors this behavior by reaching $\delta = 1.00$, its maximum possible value, indicating that there are no pairwise comparisons in which the control group obtains higher values. These results do not represent anomalies; rather, they constitute the mathematically appropriate outcome when the experimental condition uniformly outperforms the control condition across all observations.

Interpreted within the context of the task, these large effects capture the degree to which the catalog transforms participant performance. For Recall and F1, the dominance values reflect a shift from consistently incomplete solutions in G_1 to substantially broader and more accurate identification of relevant techniques in G_2. The large effects observed for Precision and Accuracy further demonstrate that the increased coverage did not introduce instability or noise; instead, the catalog improved both completeness and correctness simultaneously.

When interpreted in relation to the hypotheses defined in Section 6.1.1, the observed effect sizes provide direct empirical evidence regarding the difference between the two experimental conditions. The Vargha–Delaney A_{12} values and Cliff's δ estimates quantify the magnitude and direction of differences between G_1 and G_2 across the evaluated metrics,

indicating the extent to which the use of the catalog influences architectural decision-making outcomes.

The consistently non-trivial effect sizes observed for Recall, F1, and overall solution completeness indicate practically meaningful differences between the two groups, supporting the alternative hypothesis (H_1). In contrast, the null hypothesis (H_0), which assumes no relevant difference between conditions, is not supported by the observed effect-size magnitudes. Taken together, these results show that the catalog introduces systematic and practically relevant improvements, rather than marginal or incidental variations, in the OTA design task.

6.3 Assessing Practical Utility Through Qualitative Practitioner Feedback

6.3.1 Rationale for Assessing Practical Utility

Beyond demonstrating statistical effectiveness through controlled experimentation, this study examines the practical utility of the proposed OTA techniques catalog for practitioners involved in the design of IoT update systems. While quantitative metrics provide evidence of improvements in architectural decision quality, they do not fully capture whether an artifact is understandable, usable, or supportive of real engineering work.

For this reason, a complementary qualitative assessment was conducted to examine how practitioners perceived and engaged with the catalog during the design activity. This assessment was performed independently of the experimental measurements and provides additional evidence regarding the catalog's relevance as a decision-support artifact. Incorporating practitioner feedback enables a more complete evaluation of the catalog's applicability in realistic architectural settings.

6.3.2 Qualitative Feedback Collection Process

Qualitative feedback was collected after the experimental activity through written comments and a structured post-experiment discussion. The goal of this process was to capture practitioners' reflections on the catalog's role during architectural decision-making.

The feedback focused on two complementary aspects:

1. The perceived practical utility of the OTA techniques catalog, including its clarity, organization, and contribution to architectural reasoning.

2. The practitioners' experience when using the catalog as part of a design task, with particular attention to cognitive effort, guidance provided by the artifact, and ease of accessing relevant information.

This approach enabled the identification of recurring observations regarding how the catalog was used and how it influenced architectural decisions.

6.3.3 Perceived Practical Utility of the Techniques Catalog

Practitioners consistently described the catalog as a useful resource for informing architectural decisions regarding OTA update systems. Participants highlighted its applicability to current industrial contexts, in which IoT deployments require update mechanisms that balance security, reliability, and maintainability.

Feedback indicates that the catalog supports systematic exploration of the OTA design space by making explicit the relationships among update mechanisms, techniques, and quality attributes. Practitioners with limited prior exposure to OTA technologies reported that the catalog provided a structured reference that reduced reliance on ad-hoc reasoning. More experienced participants, in contrast, used the catalog to validate existing knowledge and to justify architectural choices in a more explicit and transparent manner.

These observations suggest that the catalog supports architectural decision-making across different levels of practitioner expertise.

6.3.4 Usefulness of Trade-Off Representation for Decision-Making

The representation of architectural trade-offs emerged as a central element in practitioners' feedback. Participants reported that the trade-off table (See Table 5.2) enabled efficient comparison of alternative techniques by explicitly linking them to quality attributes.

In several cases, practitioners relied on trade-off values to perform comparisons, prioritization, or simple aggregations aligned with scenario-specific requirements. Rather than treating quality attributes as abstract considerations, the trade-off representation enabled their use as concrete decision criteria during the design process.

This behavior reflects common engineering practices and indicates that the catalog supports decision-making by operationalizing quality concerns in a form that is directly usable during architectural design.

6.3.5 Practical Challenges Identified by Practitioners

Although the catalog was positively evaluated, practitioners also identified limitations affecting its practical use. A recurring observation concerned the density of information provided, which led some participants to focus primarily on tabular representations rather than detailed textual descriptions of individual techniques.

Participants also reported difficulties related to navigation, noting that relevant information was distributed across multiple sections of the document. Suggested improvements included introducing hyperlinks from trade-off tables to corresponding technique descriptions and providing clearer navigation support for browsing techniques.

Additionally, the use of technical acronyms and specialized terminology was identified as a potential barrier for practitioners with limited prior familiarity with OTA concepts. This feedback highlights the need for contextual support mechanisms, such as inline explanations or linked definitions, to reduce cognitive load while preserving the catalog's technical precision.

Taken together, practitioners' feedback indicates that the proposed OTA techniques catalog provides effective support for architectural decision-making beyond what can be captured through experimental metrics alone. The catalog was consistently perceived as applicable to real-world IoT and OTA design scenarios, supporting both exploratory understanding and concrete design choices.

In particular, practitioners highlighted the catalog's value as a structured reference that accommodates varying levels of prior experience. While less-experienced participants relied on the artifact to navigate the design space systematically, more-experienced practitioners used it to validate, refine, and explicitly justify their architectural decisions.

The trade-off representations played a central role in this process. By linking techniques to quality attributes in a structured and interpretable manner, the catalog enabled systematic comparison of alternatives and, in some cases, supported quantitative or semi-quantitative selection strategies aligned with scenario requirements. This observation suggests that the artifact aligns well with standard engineering practices, where architectural decisions are often guided by explicit evaluation of competing quality concerns.

At the same time, the feedback identified opportunities to improve the artifact's presentation and accessibility. Identified challenges were primarily related to navigation and information density, as well as to the use of technical acronyms and specialized terminology. Importantly, these observations point to usability and presentation refinements rather than to conceptual limitations of the catalog itself.

Overall, the qualitative evidence reinforces the interpretation that the catalog functions as a practical decision-support artifact for OTA update system design. It complements the

experimental findings by demonstrating that observed improvements in decision quality are grounded in practitioners' perceived usefulness, clarity, and applicability of the proposed solution.

6.4 Threats to Validity

This section discusses the main threats to the experiment's validity and the strategies used to mitigate them, organized according to established categories in empirical software engineering.

6.4.1 Internal Validity

Learning effects may arise because the experimental group received a brief orientation to the catalog's structure. To minimize this threat, the briefing was strictly procedural and avoided any discussion of OTA techniques or mechanisms relevant to the scenario. All participants received an equivalent clarification on the notion of "architectural technique" to ensure a shared conceptual baseline. Treatment asymmetry, inherent in the design, was controlled by ensuring that the additional information provided to the experimental group did not pre-structure the reasoning needed for the task.

Potential experimenter influence was mitigated through scripted instructions, uniform handling of questions, and the prohibition of technical clarifications during the task. Participants were assigned to groups using stratified randomization based on seniority, reducing systematic differences that could confound the results. All sessions were conducted individually and under supervision to prevent communication, dominance effects, or cross-contamination between participants.

6.4.2 External Validity

The experiment was conducted within a single company, which limits the direct generalizability of the findings to other organizational contexts. This design choice reflects a common constraint in industrial empirical studies, where access to professional practitioners and real-world design settings is inherently limited (Kitchenham and Madeyski, 2024). To mitigate this threat, the participant pool encompassed diverse areas of specialization and experience levels, reflecting the multidisciplinary composition typically found in industrial IoT engineering teams.

The distribution of technical backgrounds across groups was not controlled beyond seniority stratification and therefore may vary by chance. While this may limit the direct

generalization of the results to teams with different profile compositions, it reflects the natural heterogeneity of industrial IoT teams. Moreover, the consistent effect patterns observed across all evaluation metrics suggest that the impact of the catalog is robust to incidental differences in technical background.

In addition, the study focused on a single OTA design scenario, which does not cover the full spectrum of OTA update challenges. This scenario was deliberately constructed to represent a realistic and non-trivial case, incorporating heterogeneous devices, compatibility constraints, operational windows, security requirements, and representative update workflows commonly observed in IoT deployments. While other scenarios may emphasize different quality priorities, the selected case captures core OTA concerns that are broadly applicable across domains.

The individual nature of the task further constrains generalization to collaborative design processes, which are common in industrial practice. This choice was intentional, as it allowed the isolation of individual architectural reasoning and avoided interaction and dominance effects that could confound the comparison between experimental conditions. As a result, the conclusions should be interpreted as evidence of impact under controlled industrial conditions, capturing the direction and practical relevance of the observed effects rather than providing universal predictions for all OTA design contexts.

The experimental sample size is relatively small, which may limit statistical power and generalizability. Although comparable to similar controlled studies in software engineering, such as in (Osses et al., 2018), future replications with larger and more diverse participant groups would strengthen external validity.

6.4.3 Construct Validity

The construction of the GT and the assessment of quality attribute impacts relied on expert judgment, which introduces the potential risk of dominance, anchoring, or domain-specific bias. As shown in Table 4.7, all experts share a strong background in software architecture, reflecting the architectural scope of the study. While this may introduce a software-oriented perspective, the experts' experience also spans IoT systems, information security, and IT management, providing exposure to deployment, operational, and cross-layer concerns beyond pure software development.

Importantly, the GT and QA evaluation focus on architectural responsibilities and trade-offs rather than low-level implementation or hardware-specific optimizations. From this perspective, a common architectural background supports conceptual consistency in the interpretation of techniques and quality impacts. To mitigate individual bias, the GT definition and QA assessment followed a structured process based on independent evaluations and

moderated consensus, with unanimous agreement required for inclusion. As a result, while expert bias cannot be entirely eliminated, its influence is bounded by both the architectural focus of the study and the methodological safeguards applied.

Also, free-form responses from G_1 required mapping onto catalog techniques, which introduces potential interpretation bias. To address this, each expert performed the mapping independently, followed by a reconciliation process guided by explicit semantic criteria.

The operationalization of performance metrics (Precision, Recall, F1, Accuracy) reflects alignment with the GT rather than absolute architectural optimality. This threat is mitigated by grounding the GT in a transparent, multi-expert methodology and by aligning it with the scenario's requirements rather than a single architect's preferences.

The qualitative feedback gathered in Section 6.3 was used solely to evaluate practitioners' perceptions of the practical utility of the catalog. This feedback was not used as a measurement tool or for statistical inference; rather, it served as supplementary evidence to contextualize the experimental results. By doing so, we reduce the risk of misinterpreting constructs that could arise from mixing quantitative and qualitative evaluation purposes.

Ethical Considerations

The experimental evaluation followed established best practices for controlled experiments in software engineering, including voluntary participation, prior disclosure of study purpose and conditions, and anonymized data handling. Formal ethics approval from an ethics committee was not required because the study did not involve private or sensitive personal data, physical or psychological risks, or the recording of personal identifiers; participants performed professional design tasks in a controlled academic setting. Participants registered through a form that included the explicit consent question: *"Do you consent to the use of the data in the experiment results anonymously for the purpose of analyzing the results?"* with response options *Yes / No*, and only those selecting Yes were included.

6.4.4 Conclusion Validity

With ten participants, the study lacks statistical power for significance testing. To mitigate this limitation, the analysis relies on descriptive statistics and robust nonparametric effect-size estimators such as Vargha–Delaney A_{12} and Cliff's δ, which remain stable under small sample sizes and non-normal distributions.

To avoid unwarranted generalizations, the conclusions emphasize the strength, coherence, and directionality of the observed effects within the methodological boundaries of the study. The experimental evidence reveals a clear and systematic advantage for participants supported

by the catalog, manifested consistently across all evaluation metrics. These results constitute robust and credible evidence in favor of the hypotheses under controlled industrial conditions. While extending these findings to broader contexts requires replication across additional settings, the present experiment provides a rigorous and empirically grounded assessment of the catalog's impact on architectural decision-making and establishes a strong foundation for future replication studies.

Chapter 7

Discussion and Interpretation

The findings of this study reveal how the DeOTA-IoT catalog reshapes practitioners' reasoning about OTA update systems when quality attributes and architectural trade-offs are made explicit. The catalog does more than organize existing OTA practices; it exposes the multidimensional nature of OTA design by clarifying how each technique influences the system qualities typically required. This explicit articulation fundamentally alters the decision-making process, particularly in environments where designers must balance security, availability, performance, scalability, and energy constraints under operational and hardware heterogeneity.

The experimental results clearly illustrate this effect. Participants without the catalog relied primarily on personal experience, producing designs that were reasonable but systematically incomplete. Their tendency to omit essential techniques, as reflected in low Recall and high false-negative counts, indicates that unaided reasoning does not naturally recover the full set of responsibilities required for dependable OTA update systems.

In contrast, the catalog enabled participants to navigate the design space more comprehensively. The improvements in Recall and F1, together with more uniform coverage across mechanisms, provide empirical support for H_1 and demonstrate that the trade-off model is particularly effective in making relevant architectural concerns visible. By externalizing the effects of each technique on ten quality attributes, the catalog encourages designers to reason holistically rather than focusing narrowly on familiar areas such as authentication or secure transport. The reduced variability in G_2 further indicates that the catalog not only improves decision quality but also stabilizes it by mitigating differences in individual experience within teams.

Importantly, the effect-size analysis demonstrates that these differences are not marginal but practically meaningful, indicating that the catalog substantively influences architectural decision-making rather than merely refining existing intuitions.

The experiment also reveals a deeper insight: the presence of trade-offs does not complicate the design process; it clarifies it. Participants using the catalog were able to understand why specific techniques must be used together, why others compensate for one another's weaknesses, and why some techniques are necessary even when their benefits are indirect. This suggests that engineers intuitively appreciate quality-driven reasoning when it is explicitly articulated, reinforcing the idea that OTA design benefits from structured guidance rather than ad-hoc exploration.

Some techniques still showed patterns of misunderstanding or inconsistent selection, suggesting that further refinement or clearer descriptions could enhance the catalog's effectiveness. This is unsurprising given the technical diversity of OTA mechanisms and highlights an opportunity for future iterations of the catalog to incorporate examples, constraints, or usage scenarios to support more precise interpretation.

Although the study follows established practices for small-sample industrial experiments, the validation was conducted in a single organization and under a specific OTA scenario. Broader studies in other sectors, such as health, industrial automation, or smart mobility, may reveal how quality priorities shift the relevance of certain techniques or highlight additional trade-offs not captured in this experiment.

The trade-off model deliberately emphasizes quality attributes and does not explicitly model economic or implementation-time costs. This reflects the architectural scope of the study: while quality trade-offs can be reasoned about at a conceptual and system level, cost and effort are inherently context-sensitive and depend on organizational constraints, platform maturity, and deployment scale. Consequently, the catalog is intended to support quality-aware architectural reasoning, with cost considerations treated as a complementary, context-specific dimension that can be incorporated when sufficient situational information is available.

These quantitative improvements are consistent with the qualitative evidence reported in Section 6.3, where practitioners described the catalog as a useful and applicable decision-support artifact in realistic OTA design scenarios.

Overall, the results show that the catalog serves as a decision-support instrument that strengthens OTA design by making quality concerns explicit, reducing omissions, and supporting more balanced and consistent reasoning. These insights reinforce the catalog's role not merely as a reference but as an architectural tool that improves practitioners' approach to the inherently complex domain of OTA update systems.

Chapter 8

Conclusions and Future Work

This thesis presents a consolidated architectural foundation for designing OTA update systems in IoT, integrating five years of new evidence into a unified catalog of 34 techniques, six mechanisms, and a trade-off model that makes the quality implications of OTA decisions explicit. Through a rigorous homologation pipeline, expert-driven QA assessment, and an industrial experiment involving professional practitioners, the study demonstrates that the explicit articulation of OTA techniques supports more systematic and higher-quality architectural decisions than unaided experience alone.

Beyond individual contributions, the catalog provides a shared architectural vocabulary that can reduce ambiguity in OTA system design, promote consistency across heterogeneous IoT platforms, and strengthen the rationale for design trade-offs. The experiment confirms that practitioners benefit from this structured guidance even when operating under time constraints and diverse technological backgrounds. These findings offer initial evidence that the catalog can serve as both an analytical instrument and a practical reference for real-world IoT development.

In addition to the experimentally observed improvements in architectural decision quality, the qualitative assessment indicates that the catalog is perceived by practitioners as a practical and usable decision-support artifact for OTA update system design.

The study also highlights that the value of the catalog extends beyond enumerating techniques: it delineates the interplay between mechanisms, quality attributes, and design responsibilities, offering a foundation for more transparent, traceable, and quality-aware architectural decisions in OTA scenarios. This contribution is particularly relevant in industrial IoT environments, where device heterogeneity, connectivity limitations, and operational constraints require disciplined design practices.

Future research will broaden the empirical validation of the catalog by incorporating larger participant groups, multiple companies, and diverse IoT application domains. Multi-

organization and longitudinal studies will strengthen the catalog's generalizability and inform refinements to its mechanisms and techniques. A complementary direction is the development of an AI-driven decision-support system that operationalizes the catalog. By using its structured metadata and trade-off model, this system would provide context-aware recommendations and automate reasoning about OTA design decisions, enabling practical, large-scale adoption of the catalog in industry.

An important direction for future work is to incorporate economic and implementation-time cost considerations into the OTA techniques catalog. While the present study intentionally focuses on architectural quality attributes to support early-stage design reasoning, economic cost is a critical factor in practical engineering decisions, particularly in resource-constrained IoT environments. Future research may extend the catalog by introducing cost-related dimensions calibrated to specific organizational, platform, or deployment contexts, enabling more comprehensive trade-off analyses that combine quality attributes with economic and effort-based considerations. Such extensions could be supported through empirical studies, industrial case analyses, or parameterized cost models, complementing the quality-driven perspective adopted in this thesis.

As future work, the development of a software tool to support the recommendation of OTA update techniques is envisioned. Such a tool would leverage the DeOTA-IoT catalog and incorporate artificial intelligence–based techniques to assist designers by selecting and prioritizing OTA update techniques according to the characteristics of a given IoT system and the specific requirements of the OTA update process, including quality attributes, deployment constraints, and operational contexts. By automating the interpretation of catalog data and trade-off information through data-driven and intelligent reasoning mechanisms, the tool could provide systematic guidance during early design stages, reduce reliance on ad hoc decision-making, and facilitate the practical adoption of the proposed catalog in both academic and industrial settings.

8.1 Procedure for Expanding the DeOTA-IoT Techniques Catalog

The DeOTA-IoT catalog was built through a systematic literature-based process followed by multi-stage homologation and consolidation. To ensure the catalog remains current as new OTA update techniques emerge, it can be expanded incrementally without having to repeat the study from scratch. The expansion process preserves methodological consistency with

the original study while integrating new evidence in a controlled manner. The procedure for catalog expansion comprises the following steps:

- **Step 1: Define the Expansion Period**

  Specify the new temporal window to be incorporated into the catalog (e.g., 2026–2028). The expansion period must begin immediately after the final year covered in the current catalog version to avoid overlap.

- **Step 2: Re-execute the Literature Search**

  Repeat the original search strategy using the same digital libraries and search string, updating only the publication date filter.

  - Digital libraries: ACM Digital Library, IEEE Xplore, ScienceDirect
  - Search string: *(OTA OR "over-the-air") AND (IoT OR "internet of things")*
  - Date filter: new expansion period only

  Maintaining the same search configuration ensures methodological comparability with the original dataset.

- **Step 3: Apply the Same Inclusion and Exclusion Criteria**

  Screen the retrieved studies using the original inclusion and exclusion criteria defined in 4.2.3.

  This guarantees consistency in the definition of what qualifies as an OTA update technique.

- **Step 4: Extract Candidate Techniques from New Studies**

  From the filtered studies, extract all OTA-related procedures, approaches, or methods that:

  1. Implement an OTA update responsibility
  2. Affect OTA behavior or properties
  3. Can be expressed as a technique

  Each extracted technique should be recorded with:

  1. Source reference
  2. Original description

3. Target mechanism (if identifiable)
4. Context (device, network, domain)

The result of this step is a set of candidate techniques for expansion.

- **Step 5: Normalize and Consolidate the New Techniques List**

  Before integration, standardize terminology and granularity:

  1. Merge duplicates across new studies
  2. Split composite techniques if needed
  3. Align naming with catalog conventions

  This produces a clean list of new techniques.

- **Step 6: Homologate Against the Existing Catalog**

  Compare each candidate technique with the existing 34 catalog techniques to determine its status.

  For each candidate:

  1. Equivalent: merge with existing technique
  2. Variant: extend existing technique description
  3. Refinement: add as sub-variant
  4. Novel: add as new technique

  Homologation criteria:

  1. Functional goal
  2. OTA responsibility
  3. Operational procedure
  4. Quality-attribute effects

  This step ensures conceptual continuity with the catalog.

- **Step 7: Assign Mechanisms to New or Updated Techniques**

  For techniques classified as novel or refined, determine the OTA mechanism they realize using the existing mechanisms taxonomy. Each technique must map to at least one mechanism.

  This preserves the architectural structure of the catalog.

- **Step 8: Update Quality-Attribute Trade-off Profiles**

  Evaluate new or modified techniques using the same Likert-scale impact model applied in the original catalog.

  Attributes:

  - Security
  - Reliability
  - Availability
  - Scalability
  - Performance
  - Energy management
  - Interoperability
  - Privacy
  - Flexibility
  - Evolvability

  Expert assessment or literature-based justification should be recorded.

- **Step 9: Integrate into the Catalog Dataset**

  Add homologated techniques and updated profiles to the master catalog through the following actions:

  1. Assign unique technique IDs
  2. Update mechanism groupings
  3. Revise totals and distributions
  4. Update tables and diagrams

- **Step 10: Document Version and Expansion Scope**

  Record the following information:

  1. Expansion period
  2. Number of studies screened
  3. Number of techniques added/merged
  4. Changes to existing techniques

  This enables traceability across catalog versions.

Result of the Expansion Process:

Following the specified procedure yields an updated DeOTA-IoT catalog that:

1. Preserves methodological consistency
2. Maintains homologation integrity
3. Incorporates new OTA evidence
4. Retains architectural structure
5. Supports longitudinal evolution

Thus, the catalog can be incrementally maintained as the OTA update domain evolves without repeating the full study lifecycle.

References

Abdallah, M., Jaber, T., Alabwaini, N., and Alnabi, A. A. (2019). A proposed quality model for the internet of things systems. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pages 23–27.

Abdelfadeel, K., Farrell, T., McDonald, D., and Pesch, D. (2020). How to make firmware updates over lorawan possible. In *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pages 16–25.

Ahmed, B. S., Bures, M., Frajtak, K., and Cerny, T. (2019). Aspects of quality in internet of things (iot) solutions: A systematic mapping study. *IEEE Access*, 7:13758–13780.

Ali, M., Saleem, Y., Hina, S., and Shah, G. A. (2025). Ddosvit: Iot ddos attack detection for fortifying firmware over-the-air (ota) updates using vision transformer. *Internet of Things*, 30:101527.

Andrade, C. E., Byers, S. D., Gopalakrishnan, V., Halepovic, E., Poole, D. J., Tran, L. K., and Volinsky, C. T. (2019). Scheduling software updates for connected cars with limited availability. *Applied Soft Computing*, 82:105575.

Anedda, M., Floris, A., Girau, R., Fadda, M., Ruiu, P., Farina, M., Bonu, A., and Giusto, D. D. (2023). Privacy and security best practices for iot solutions. *IEEE Access*, 11:129156–129172.

Antonino, P. O., Capilla, R., Pelliccione, P., Schnicke, F., Espen, D., Kuhn, T., and Schmid, K. (2022). A quality 4.0 model for architecting industry 4.0 systems. *Advanced Engineering Informatics*, 54:101801.

Aoki, N., Takefusa, A., Ishikawa, Y., Ono, Y., Sakane, E., and Aida, K. (2024). Zt-ota update framework for iot devices toward zero trust iot. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 2195–2202.

Arakadakis, K., Charalampidis, P., Makrogiannakis, A., and Fragkiadakis, A. (2021). Firmware over-the-air programming techniques for iot networks - a survey. *ACM Comput. Surv.*, 54(9).

Arakadakis, K., Karamolegkos, N., and Fragkiadakis, A. (2022). Dfinder—an efficient differencing algorithm for incremental programming of constrained iot devices. *Internet of Things*, 17:100482.

Arakaki, R., Hayashi, V. T., and Ruggiero, W. V. (2020). Available and fault tolerant iot system: Applying quality engineering method. In *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pages 1–6.

Asokan, N., Nyman, T., Rattanavipanon, N., Sadeghi, A., and Tsudik, G. (2018). ASSURED: Architecture for secure software update of realistic embedded devices. arXiv preprint.

Babic, S. and contributors (2025). Swupdate: Software update for embedded systems. https://sbabic.github.io/swupdate/swupdate.html. Open-source project, version as of May 2025; Accessed on: 2025-11-03.

Banijamali, A., Heisig, P., Kristan, J., Kuvaja, P., and Oivo, M. (2019). Software architecture design of cloud platforms in automotive domain: An online survey. In *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)*, pages 168–175.

Banijamali, A., Pakanen, O.-P., Kuvaja, P., and Oivo, M. (2020). Software architectures of the convergence of cloud computing and the internet of things: A systematic literature review. *Information and Software Technology*, 122:106271.

Bas, J. and Dowhuszko, A. A. (2021). On the use of nb-iot over geo satellite systems with time-packed optical feeder links for over-the-air firmware, software updates of machine-type terminals. *Sensors*, 21.

Bass, L., Clements, P., and Kazman, R. (2021). *Software Architecture in Practice*. Addison-Wesley, Boston, MA, USA, 4th edition. 4th ed.

Batyrshin, I. Z., Monroy-Tenorio, F., Gelbukh, A., Villa-Vargas, L. A., Solovyev, V., and Kubysheva, N. I. (2017). Bipolar rating scales: A survey and novel correlation measures based on nonlinear bipolar scoring functions. *Acta Polytechnica Hungarica*, 14:33–57.

Bauwens, J., Ruckebusch, P., Giannoulis, S., Moerman, I., and Poorter, E. D. (2020). Over-the-air software updates in the internet of things: An overview of key principles. *IEEE Communications Magazine*, 58(2):35–41.

Bradley, C. and Barrera, D. (2023). Escaping vendor mortality: A new paradigm for extending iot device longevity. In *2023 New Security Paradigms Workshop*, NSPW '23, pages 1–16, New York, NY, USA. Association for Computing Machinery.

Brown, S. and Sreenan, C. J. (2013). Software updating in wireless sensor networks: A survey and lacunae. *Journal of Sensor and Actuator Networks*, 2:717–760.

Bures, M., Bellekens, X., Frajtak, K., and Ahmed, B. S. (2020). A comprehensive view on quality characteristics of the iot solutions. In José, R., Van Laerhoven, K., and Rodrigues, H., editors, *3rd EAI International Conference on IoT in Urban Space*, pages 59–69, Cham. Springer International Publishing.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*, volume 1 of *Pattern-Oriented Software Architecture*. Wiley.

Cadavid, H., Andrikopoulos, V., and Avgeriou, P. (2020). Architecting systems of systems: A tertiary study. *Information and Software Technology*, 118:106202.

Catalano, J. (2021). Lorawan firmware update over-the-air (fuota). *Journal of ICT Standardization*, 9(1):21–34.

Charalampidis, P., Makrogiannakis, A., Karamolegkos, N., Papadakis, S., Charalambakis, Y., Kamaratakis, G., and Fragkiadakis, A. (2022). A flexible compilation-as-a-service and remote-programming-as-a-service platform for iot devices. *Internet of Things*, 20:100617.

Chhiba, L., Elhadi, S., Marzak, A., and Sidqui, M. (2022). Evaluating iot-based healthcare architecture with quality factors. In Motahhir, S. and Bossoufi, B., editors, *Digital Technologies and Applications*, pages 302–311, Cham. Springer International Publishing.

Chien, H.-Y. and Wang, N.-Z. (2022). A novel mqtt 5.0-based over-the-air updating architecture facilitating stronger security. *Electronics*, 11(23).

Choi, S. and Lee, J.-H. (2020). Blockchain-based distributed firmware update architecture for iot devices. *IEEE Access*, 8:37518–37525.

Christensen, H. B., Hansen, K. M., and Lindstrøm, B. (2010). asqa: Architectural software quality assurance: Software architecture at work – technical report #5. Accessed: 2025-12-31.

Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin*, 114(3):494–509.

Crowther, K. G., Upadrashta, R., and Ramachandra, G. (2022). Securing over-the-air firmware updates (fota) for industrial internet of things (iiot) devices. In *2022 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–8.

Da Silva, G. F. P., Costa, D. G., and De Jesus, T. C. (2023). A secure ota approach for flexible operation of emergency detection units in smart cities. In *2023 IEEE International Smart Cities Conference (ISC2)*, pages 01–07.

De, B. (2017). *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization*. Apress, 1st edition.

de Sousa, M. J. B., Gonzalez, L. F. G., Ferdinando, E. M., and Borin, J. F. (2022). Over-the-air firmware update for iot devices on the wild. *Internet of Things*, 19:100578.

El Jaouhari, S. (2022). Toward a secure firmware ota updates for constrained iot devices. In *2022 IEEE International Smart Cities Conference (ISC2)*, pages 1–6.

El Jaouhari, S. and Bouvet, E. (2022). Secure firmware over-the-air updates for iot: Survey, challenges, and discussions. *Internet of Things*, 18:100508.

Elhanashi, A., Dini, P., Saponara, S., and Zheng, Q. (2024). Advancements in tinyml: Applications, limitations, and impact on iot devices. *Electronics*, 13.

Empl, P., Schlette, D., Zupfer, D., and Pernul, G. (2022). Soar4iot: Securing iot assets with digital twins. In *17th International Conference on Availability, Reliability and Security*, ARES '22, New York, NY, USA. Association for Computing Machinery.

Fagan, M., Megas, K., Scarfone, K., and Smith, M. (2020). Foundational cybersecurity activities for iot device manufacturers. https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8259.pdf. Accessed on: 2024-11-22.

Felderer, M. and Travassos, G. (2020). *Contemporary Empirical Methods in Software Engineering*. Springer.

Finstad, K. (2010). The system usability scale and non-normality: A practical examination. *Journal of Usability Studies*, 5(3):117–123.

Foundation, E. (2025). Eclipse hawkbit - update server. https://eclipse.dev/hawkbit/. Accessed on: 2025-11-03.

Gu, J., Lee, S.-S., and Kang, H. (2024). Dsme-fota: Firmware over-the-air update framework for ieee 802.15.4 dsme mac to enable large-scale multi-hop industrial iot networks. *Internet of Things*, 27:101239.

Halder, S., Ghosal, A., and Conti, M. (2019). Secure ota software updates in connected vehicles: A survey. arXiv preprint.

Harrison, N. B. and Avgeriou, P. (2010). How do architecture patterns and tactics interact? a model and annotation. *Journal of Systems and Software*, 83(10):1735–1758.

Hayati, N. (2024). The design of security framework for lorawan fuota. *Journal of Electrical Technology UMY*, 7(2):82–88.

He, X., Alqahtani, S., Gamble, R., and Papa, M. (2019). Securing over-the-air iot firmware updates using blockchain. In *International Conference on Omni-Layer Intelligent Systems*, COINS '19, pages 164–171, New York, NY, USA. Association for Computing Machinery.

Heins, K. (2022). *NB-IoT Use Cases and Devices: Design Guide*. Springer, Cham, Switzerland.

Hernández-Ramos, J. L., Baldini, G., Matheu, S. N., and Skarmeta, A. (2020). Updating iot devices: challenges and potential approaches. In *2020 Global Internet of Things Summit (GIoTS)*, pages 1–5.

Hidaka, H., editor (2018). *Embedded Flash Memory for Embedded Systems: Technology, Design for Sub-systems, and Innovations*. Integrated Circuits and Systems. Springer.

ISO/IEC (2001). ISO/IEC 9126-1: Software engineering – product quality – part 1: Quality model. International Standard. Geneva, Switzerland.

ISO/IEC (2023). Systems and software engineering — systems and software quality requirements and evaluation (square) — product quality model. https://www.iso.org/standard/78176.html. International Standard, Ref. no. 78176.

Jang, H.-C. and Tsai, C.-Y. (2021). Innovative fota-based software update for iot mesh network. In *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0577–0582.

Jaouhari, S. E. and Bouvet, E. (2022). Toward a generic and secure bootloader for iot device firmware ota update. In *2022 International Conference on Information Networking (ICOIN)*, pages 90–95.

Johnson, R. (2025). *MQTT Protocol in Practice: Definitive Reference for Developers and Engineers*. HiTeX Press.

Jones, M., Bradley, J., and Sakimura, N. (2015a). Json web signature (jws). https://www.rfc-editor.org/rfc/rfc7515.html. Standards Track.

Jones, M. B., Bradley, J., and Sakimura, N. (2015b). Json web token (jwt). https://www.rfc-editor.org/info/rfc7519.

K, S., S, J., Kovilpillai J, J. A., Santhosh S, S., and R, S. (2024). Efficient implementation of firmware over-the-air update using raspberry pi 4 and stm32f103c8t6. In *2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–6.

Kazman, R., Bass, L., Abowd, G., and Webb, M. (1994). SAAM: A method for analyzing the properties of software architectures. pages 81–90.

Kazman, R., Klein, M., and Clements, P. (2000). Atam: Method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, USA. Accessed: 2025-12-31.

Kerliu, K., Ross, A., Tao, G., Yun, Z., Shi, Z., Han, S., and Zhou, S. (2019). Secure over-the-air firmware updates for sensor networks. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, pages 97–100.

Kesari, A., Srivastav, A., Singh, A., Bhanotia, H., and Rather, M. A. (2025). Firmware in flight: A deep dive into ota update mechanisms for iot. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 13(V):1254–1268.

Khezemi, N., Baptiste Minani, J., Sabir, F., Moha, N., Guéhéneuc, Y.-G., and El Boussaidi, G. (2024). A systematic literature review of iot system architectural styles and their quality requirements. *IEEE Internet of Things Journal*, 11(23):37599–37616.

Kim, M. (2016). A quality model for evaluating iot applications. *International Journal of Computer and Electrical Engineering*, 8(1):66–76.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., and Linkman, S. (2009). Systematic literature reviews in software engineering: A systematic literature review. *Information and Software Technology*, 51(1).

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Kitchenham, B. and Madeyski, L. (2024). Recommendations for analysing and meta-analysing small sample size software engineering experiments. *Empirical Software Engineering*, 29(6):137.

Kozlowski, T., Noran, O., and Trevathan, J. (2023). Designing an evaluation framework for iot environmental monitoring systems. *Procedia Computer Science*, 219:220–227. CENTERIS – International Conference on ENTERprise Information Systems / ProjMAN – International Conference on Project MANagement / HCist – International Conference on Health and Social Care Information Systems and Technologies 2022.

Krishnan, P., Jain, K., Poojara, S. R., Srirama, S. N., Pandey, T., and Buyya, R. (2024). esim and blockchain integrated secure zero-touch provisioning for autonomous cellular-iots in 5g networks. *Computer Communications*, 216:324–345.

Kubaščík, M., Tupý, I. A., Šumský, J., and Bača, T. (2024). Ota firmware updates on esp32 based microcontrolers. In *2024 IEEE 17th International Scientific Conference on Informatics (Informatics)*, pages 185–189.

Kuppusamy, T. K., DeLong, L. A., and Cappos, J. (2018). Uptane: Security and customizability of software updates for vehicles. *IEEE Vehicular Technology Magazine*, 13(1):66–73.

Labs, P. (2025). Interplanetary file system (ipfs) — content addressed, versioned, peer-to-peer file system. https://ipfs.tech/. Accessed on: 2025-11-03.

Lau, H., Liu, L., and Chan, K. (2025). *Introduction to Wireless System Design: From Circuits to Web-based Applications*. John Wiley & Sons, Hoboken, NJ, USA.

LLC, G. (2025). Flatbuffers: Memory efficient serialization library. https://google.github.io/flatbuffers/. Version: latest commit as of 2025-11-22; Apache 2.0 license; Accessed on: 2025-11-03.

Mahfoudhi, F., Sultania, A. K., and Famaey, J. (2022). Over-the-air firmware updates for constrained nb-iot devices. *Sensors*, 22(19).

Malumbres, V., Saldana, J., Berné, G., and Modrego, J. (2024). Firmware updates over the air via lora: Unicast and broadcast combination for boosting update speed. *Sensors*, 24(7).

Matrosov, A., Rodionov, E., and Bratus, S. (2019). *Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats*. No Starch Press, San Francisco, CA, USA. First edition.

Minerva, R., Biru, A., and Rotondi, D. (2015). Towards a definition of the internet of things (iot). https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf. Accessed on: 2024-11-22.

Mirsky, Y., Golomb, T., and Elovici, Y. (2020). Lightweight collaborative anomaly detection for the iot using blockchain. *Journal of Parallel and Distributed Computing*, 145:75–97.

Mishra, A., Alzoubi, Y. I., and Gavrilovic, N. (2024). Quality attributes of software architecture in iot-based agricultural systems. *Smart Agricultural Technology*, 8:100523.

Moaven, S. and Habibi, J. (2020). A fuzzy-ahp-based approach to select software architecture based on quality attributes (fassa). *Knowledge and Information Systems*, 62(5):4569–4597. Accessed: 2025-12-31.

Moran, B., Tschofenig, H., Brown, D., and Meriac, M. (2021). Rfc 9019: A firmware update architecture for internet of things. https://www.rfc-editor.org/info/rfc9019.

Muñoz, A. and Fernandez, E. B. (2020). Tpm, a pattern for an architecture for trusted computing. In *European Conference on Pattern Languages of Programs 2020*, EuroPLoP '20, New York, NY, USA. Association for Computing Machinery.

National Telecommunications and Information Administration (NTIA) (2017). Catalog of existing iot security standards. Existing Standards, Tools and Initiatives Working Group (WG1), September 12, 2017.

Nguyen, H. D., Sommer, N. L., and Mahéo, Y. (2024). Over-the-air firmware update in lorawan networks: A new module-based approach. *Procedia Computer Science*, 241:154–161. 19th International Conference on Future Networks and Communications, 21th International Conference on Mobile Systems and Pervasive Computing, 14th International Conference on Sustainable Energy Information Technology.

Nikbakht Bideh, P. and Gehrmann, C. (2022). Rosym: Robust symmetric key based iot software upgrade over-the-air. In *4th Workshop on CPS & IoT Security and Privacy*, CPSIoTSec '22, pages 35–46, New York, NY, USA. Association for Computing Machinery.

Open Mobile Alliance (2025). Lightweight machine to machine (lwm2m) specification – introduction. https://www.openmobilealliance.org/specifications/lwm2m. Accessed on: 2025-11-03.

Orellana, C. and Astudillo, H. (2025). Architectural tactics and trade-offs for confidentiality. *Journal of Systems and Software*, 226:112433.

Orellana, C., Astudillo, H., and Fernandez, E. B. (2022a). A pattern for a secure actuator node. In *26th European Conference on Pattern Languages of Programs*, EuroPLoP '21, New York, NY, USA. Association for Computing Machinery.

Orellana, C., Cereceda-Balic, F., Solar, M., and Astudillo, H. (2024). Enabling design of secure iot systems with trade-off-aware architectural tactics. *Sensors*, 24.

Orellana, C., Fernandez, E. B., and Astudillo, H. (2022b). A pattern for a secure sensor node. In *27th Conference on Pattern Languages of Programs*, PLoP '20, USA. The Hillside Group.

Orellana, C., Villegas, M. M., and Astudillo, H. (2019). Assessing architectural patterns trade-offs using moment-based pattern taxonomies. pages 1–8.

Osses, F., Márquez, G., Villegas, M., Orellana, C., Visconti, M., and Astudillo, H. (2018). Security tactics selection poker (TaSPeR): a card game to select security tactics to satisfy security requirements. In *12th European Conference on Software Architecture: Companion Proceedings (ECSA 2018)*, ECSA '18, New York, NY, USA. Association for Computing Machinery.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M.-M. L., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., McGuinness, L. A., Stewart, L., Thomas, J., Tricco, A., Welch, V. A., Whiting, P., and Moher, D. (2021). The prisma 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*, 372:n71. Accessed: 2025-11-07.

Park, C.-Y., Lee, S.-J., and Lee, I.-G. (2025). Secure and lightweight firmware over-the-air update mechanism for internet of things. *Electronics*, 14(8).

Parker, K. P. (2016). *The Boundary-Scan Handbook*. Springer, Cham, Switzerland, 4 edition.

Preston, C. C. and Colman, A. M. (2000). Optimal number of response categories in rating scales: Reliability, validity, discriminating power, and respondent preferences. *Acta Psychologica*, 104(1):1–15.

PRISMA (2024). Prisma 2020 checklist. https://www.prisma-statement.org/prisma-2020-checklist. Accessed on: 2024-06-20.

Rajmohan, T., Nguyen, P. H., and Ferry, N. (2022). A decade of research on patterns and architectures for IoT security. *Cybersecurity*, 5:2.

Regenscheid, A. (2018). Platform firmware resiliency guidelines. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-193.pdf. Federal agency guidelines; freely available.

Romano, J., Kromrey, J. D., Coraggio, J., and Skowronek, J. (2006). Appropriate statistics for ordinal level data: Should we really be using t-test and cohen's d for evaluating group differences on the nsse and other surveys?. In *Annual Meeting of the Florida Association of Institutional Research*, Cocoa Beach, Florida.

Sangster, R. L., Willits, F. K., Saltiel, J., Lorenz, F. O., and Rockwood, T. H. (2001). The effects of numerical labels on response scales. Accessed on: 2025-04-22.

Schaad, J. (2022). Rfc 9052: Cbor object signing and encryption (cose): Structures and process.

Sharma, N., Awasthi, L. K., Mangla, M., Sharma, K. P., and Kumar, R., editors (2022). *Cyber-Physical Systems: A Comprehensive Guide*. Chapman & Hall, CRC Cyber-Physical Systems. CRC Press, Taylor & Francis Group, New York, USA. 1st edition.

Shi, T., Wang, R., Zhang, D., Jiao, W., and Xie, B. (2013). Quality driven design of program frameworks for intelligent sensor applications. In *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, volume 1, pages 442–449.

Shin, Y. and Jeon, S. (2024). Mqtree: Secure ota protocol using mqtt and merkletree. *Sensors*, 24(5).

Siriwardena, P. (2019). *Advanced API Security: OAuth 2.0 and Beyond*. Apress.

Soffar, H. (2024). *AUTOSAR Fundamentals and Applications: Establishing a Solid Foundation for Automotive Software Design with AUTOSAR*. Packt Publishing. 1st ed.

Sousa, M. and Borin, J. (2023). Design and evaluation of a method for over-the-air firmware updates for iot devices. In *36th Thesis and Dissertation Contest*, pages 80–87, Porto Alegre, RS, Brasil. SBC.

Srinivas, A. K., Vikram, D., Sharma, S., and Kumar lenka, R. (2022). Deployment automation for blockchain enabled iomt. In *2022 OITS International Conference on Information Technology (OCIT)*, pages 1–4.

Srivastava, A. K., CS, K., Lilaramani, D., R, R., and Sree, K. (2021). An open-source swupdate and hawkbit framework for ota updates of risc-v based resource constrained devices. In *2021 2nd International Conference on Communication, Computing and Industry 4.0 (C2I4)*, pages 1–6.

Stallings, W. and Brown, L. (2017). *Computer Security: Principles and Practice, Fourth Edition*. Pearson Education, Boston, MA, USA. 4th ed.

Sudharsan, B., Breslin, J. G., Tahir, M., Intizar Ali, M., Rana, O., Dustdar, S., and Ranjan, R. (2022). Ota-tinyml: Over the air deployment of tinyml models and execution on iot devices. *IEEE Internet Computing*, 26(3):69–78.

Sun, C., Xing, R., Wu, Y., Zhou, G., Zheng, F., and Hu, D. (2021). Design of over-the-air firmware update and management for iot device with cloud-based restful web services. In *2021 China Automation Congress (CAC)*, pages 5081–5085.

Sun, Z., Ni, T., Yang, H., Liu, K., Zhang, Y., Gu, T., and Xu, W. (2023). Flora: Energy-efficient, reliable, and beamforming-assisted over-the-air firmware update in lora networks. In *22nd International Conference on Information Processing in Sensor Networks*, IPSN '23, pages 14–26, New York, NY, USA. Association for Computing Machinery.

T. Bray, E. (2017). The javascript object notation (json) data interchange format. https://datatracker.ietf.org/doc/html/rfc8259. Standards Track, Internet Standard.

Teck Khieng, D. H., Xie, Y.-Z., Zhang, J.-C., and Huang, N.-F. (2023). A long distance low bandwidth firmware update process for lpwan - taking lorap2p+ as example. In *2023 International Conference on Information Networking (ICOIN)*, pages 646–651.

Temkar, R. and Bhaskar, A. (2021). Quality assurance of iot based home automation application using modified iso/iec 25010. *International Journal of Engineering Trends and Technology*, 69(2):92–101.

Thakur, P., Bodade, V., Achary, A., Addagatla, M., Kumar, N., and Pingle, Y. (2019). Universal firmware upgrade over-the-air for iot devices with security. In *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 27–30, New Delhi, India.

Thantharate, A., Beard, C., and Kankariya, P. (2019). Coap and mqtt based models to deliver software and security updates to iot devices over the air. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1065–1070.

Vargha, A. and Delaney, H. D. (2000). A critique and improvement of the "cl" common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132.

Villegas, M. M. and Astudillo, H. (2020). Ota updates mechanisms: A taxonomy and techniques catalog. In *XXI Simposio Argentino de Ingeniería de Software (ASSE), JAIIO 49*, pages 139–158.

Villegas, M. M., Orellana, C., and Astudillo, H. (2019). A study of over-the-air (ota) update systems for cps and iot operating systems. In *13th European Conference on Software Architecture - Volume 2 (ECSA '19)*, pages 269–272, New York, NY, USA. Association for Computing Machinery.

Wahlström, E., Bormann, C., and Selander, G. (2019). Object security for constrained restful environments (oscore). https://www.rfc-editor.org/rfc/rfc8613.html. Standards Track.

Washizaki, H., Ogata, S., Hazeyama, A., Okubo, T., Fernandez, E. B., and Yoshioka, N. (2020). Landscape of architecture and design patterns for iot systems. *IEEE Internet of Things Journal*, 7(10):10091–10101.

Wei, W., Pan, C., Islam, S., Banerjee, J., Palanisamy, S., and Xie, M. (2025). Intermittent ota code update framework for tiny energy harvesting devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 44(1):77–90.

Wilson, Y. and Hingnikar, A. (2019). *Solving Identity Management in Modern Applications: Demystifying OAuth 2.0, OpenID Connect, and SAML 2.0*. Apress.

Witanto, E. N., Oktian, Y. E., Kumi, S., and Lee, S.-G. (2019). Blockchain-based ocf firmware update. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1248–1253.

Witanto, E. N., Oktian, Y. E., Lee, S.-G., and Lee, J.-H. (2020). A blockchain-based ocf firmware update for iot devices. *Applied Sciences*, 10(19).

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer.

Yiu, J. (2020). *Definitive Guide to Arm® Cortex®-M23 and Cortex-M33 Processors*. Newnes, 1st edition.

Yohan, A., Lo, N.-W., and Santoso, L. P. (2025). A robust and efficient blockchain-based framework for updating firmware in iot environments. *Peer-to-Peer Networking and Applications*, 18(4):207.

Yılmaz, R. and Buzluca, F. (2024). A fuzzy logic-based quality model for identifying microservices with low maintainability. *Journal of Systems and Software*, 216:112143.

Zhang, Q., He, Y., Xiao, Y., Zhang, X., and Song, C. (2025). Ota-key: Over-the-air key management for flexible and reliable iot device provision. *IEEE Transactions on Network and Service Management*, 22(2):2106–2119.

Zyrianoff, I., Sciullo, L., Gigli, L., Trotta, A., Kamienski, C., and Di Felice, M. (2024). An over the air software update system for iot microcontrollers based on webassembly. In *2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, pages 331–338.

Techniques Catalog for Over-The-Air (OTA) Update Systems Design for Internet of Things (IoT) Systems

Written by: Mónica M.Villegas

V2.0
December 22, 2025

Table of Contents

Table of Abbreviations

Abbreviation	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
COSE	CBOR Object Signing and Encryption
CRC	Cyclic Redundancy Check
CRUD	Create, Read, Update, Delete
DTLS	Datagram Transport Layer Security
ECDSA	Elliptic Curve Digital Signature Algorithm
FlatBuffer	Efficient cross-platform serialization format
GPS	Global Positioning System
HTTP(S)	Hypertext Transfer Protocol (Secure)
IAM	Identity and Access Management
IoT	Internet of Things
IPFS	InterPlanetary File System
JTAG	Joint Test Action Group (debug interface)
JOSE	JSON Object Signing and Encryption
JSON	JavaScript Object Notation
JWT	JSON Web Token
LwM2M	Lightweight Machine-to-Machine
MFA	Multi-Factor Authentication
MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrowband IoT
NVM	Non-Volatile Memory
OEM	Original Equipment Manufacturer
ONNX	Open Neural Network Exchange
OS	Operating System
OSCORE	Object Security for Constrained RESTful Environments
OTP	One-Time Password
OTA	Over-the-Air
P2P	Peer-to-Peer
PUF	Physically Unclonable Function
REST	Representational State Transfer

RoT	Root of Trust
RSA	Rivest–Shamir–Adleman (encryption)
SBSFU	Secure Boot and Secure Firmware Update
SIM	Subscriber Identity Module
SWD	Serial Wire Debug
SWU (.swu)	Software Update Package format
TEE	Trusted Execution Environment
TFLite	TensorFlow Lite
TLS	Transport Layer Security
TPM	Trusted Platform Module

Introduction

This document contains a techniques catalog for each of the six mechanisms that set up an OTA Update System for IoT systems. Such mechanisms can be used to design OTA update systems for IoT. To this end, a mechanism is an architectural building block that encapsulates a specific responsibility within the update process, and a technique denotes the concrete procedure by which a mechanism is realized. In Figure 1, the OTA Update System Mechanisms are shown and described. Figure 2 presents an example of an OTA Update system, considering both Cloud (Backend) and IoT device operations, and showing where the mechanisms are present in the OTA system flow.

IoT architects, engineers, and practitioners can use the techniques for designing OTA update systems by selecting the required ones and covering at least one of each mechanism. Each technique comprises a title, a description, a list of general use cases, a technique workflow diagram illustrating its role in an implementation, and an assessment of its impact on Quality Attributes (QAs). Table 1 presents the quality attributes (QAs) associated with IoT systems. In each technique, each QA is qualified using a 5-point Likert-style bipolar scale, as shown in Table 2, based on its impact. Table 3 presents the names of all techniques in a synthesized format, providing a preview of those detailed later. Table 4 also presents, in a synthesized manner, the techniques and their impact on the QAs.

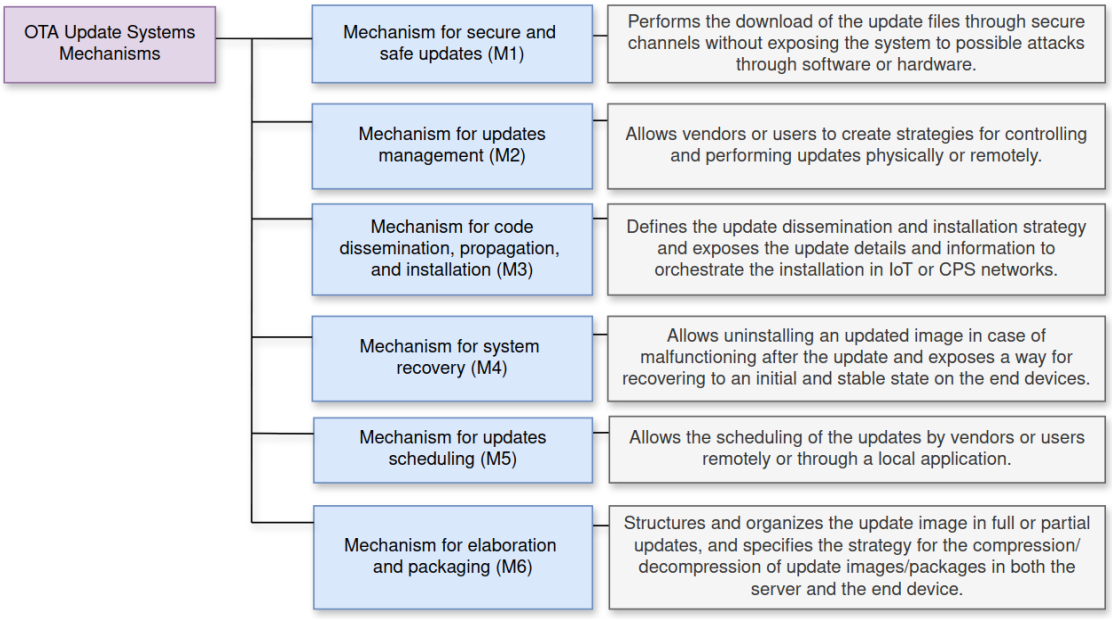


Figure 1. Description of the six mechanisms that set up an OTA update system for IoT.

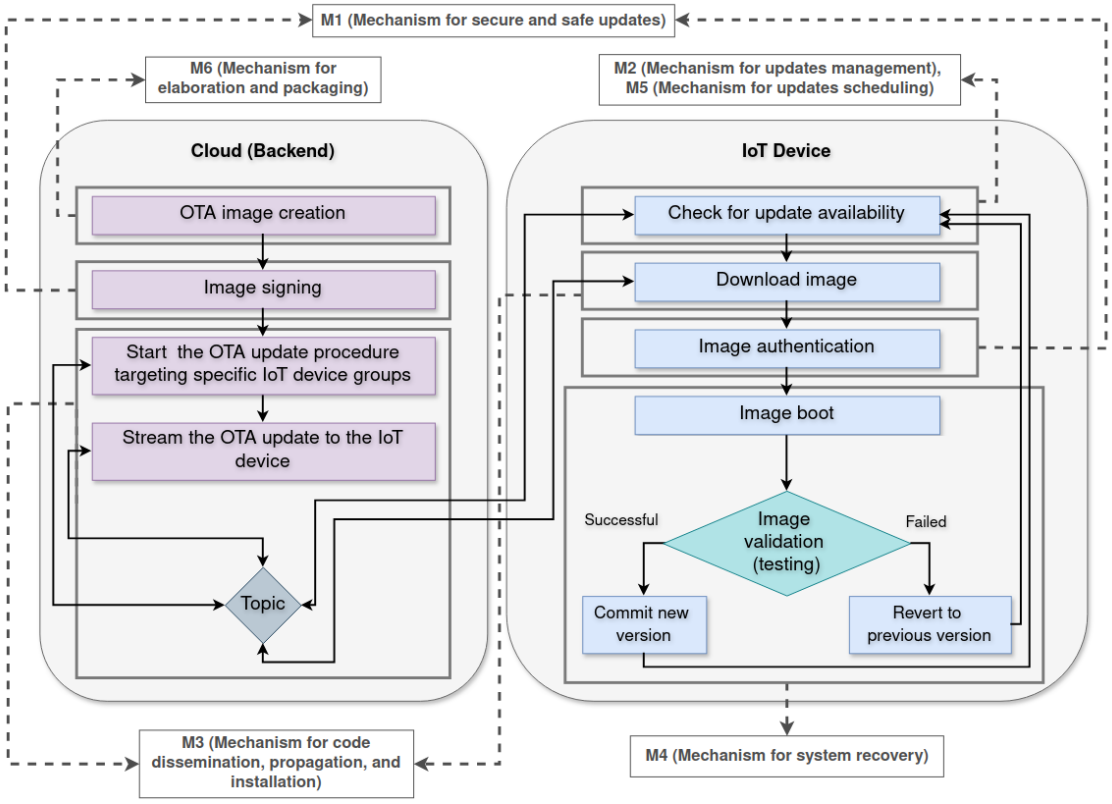


Figure 2. Example of an OTA Update System for IoT, in which the techniques are inside the gray rectangles, and the mechanisms they are associated with are shown as connected.

Table 1. IoT System Quality Attributes [1] and ther definition.

Quality Attributes (QA) in Software Architecture		
QA ID	**QA**	**Description**
QA1	Security	Ability of a system to protect itself from unauthorized access, modifications, or attacks
QA2	Scalability	Ability of a system to handle increased load or demand without degrading performance
QA3	Performance	Ability of a system to meet the required speed, responsiveness, and resource usage
QA4	Availability	Proportion of time that a system is functional and accessible
QA5	Interoperability	Ability of a system to exchange data and services with heterogeneous devices, platforms, and software components using standardized interfaces, protocols, or formats, requiring minimal adaptation or integration effort.
QA6	Reliability	Ability of a system to consistently perform its required functions under specified conditions
QA7	Privacy	Ability to safeguard personal and sensitive information collected by devices, ensuring it is only used for its intended purpose
QA8	Energy management	Ability to use energy efficiently across devices, ensuring optimal performance while minimizing power consumption
QA9	Flexibility	Ability of a system to adapt to future requirements or changes in environment with minimal disruption
QA10	Evolvability	Ability of a system to accommodate functional, technological, or operational changes over time with limited architectural restructuring, low implementation effort, and controlled impact on existing system behavior.

[1] N. Khezemi, J. B. Minani, F. Sabir, N. Moha, Y.-G. Guéhéneuc, and G. El Boussaidi, “A Systematic Literature Review of IoT System Architectural Styles and Their Quality Requirements,” IEEE Internet of Things Journal, vol. 11, no. 23, pp. 37599–37616, 2024. doi: 10.1109/JIOT.2024.3435496

Table 2. 5-point Likert-style bipolar scale for to Qualify each Quality Attribute (QA) Impact[23].

Level	Descrpition	Example
-2	Strong Negative Impact	Blockchain validation severely reduces scalability
-1	Slight Negative Impact	Encryption slightly increases energy consumption
0	Neutral (No Impact)	Metadata format doesn't affect privacy
+1	Slight Positive Impact	Incremental update modestly improves performance
+2	Strong Positive Impact	Secure boot ensures firmware authenticity

Table 3. Techniques Names Summary.

Technique ID	Technique Name
T1.1	Firmware Integrity, Authenticity, and Installation Verification
T1.2	Secure Payload Encryption and End-to-End Communication
T1.3	Trusted Execution and Runtime Protection
T1.4	Key, Credential, and Identity Management
T1.5	Decentralized and Blockchain-Based Verification
T1.6	Compatibility and Threat Prevention
T1.7	Protocol, Debug Interface, and Request Security
T1.8	Installation and Runtime Security Assurance
T2.1	OTA Orchestration and Management Interfaces
T2.2	Role-Based Access and Permissions
T2.3	Workflow Coordination and Traceability
T2.4	Device Classification and Delivery Mode Selection
T2.5	User Awareness and Consent Management
T3.1	OTA Protocols, Delivery Models, and Channel Selection
T3.2	Manifest-Driven and Policy-Based Update Control
T3.3	Reliable Transfer and Execution Resilience
T3.4	Pre-Deployment Validation and Optimization
T3.5	Lightweight Incremental Update Application
T3.6	Secure Transfer, Authentication, and Installation Readiness

[2] Demin, D.; Makhalova, E.; Batyrshin, I. Bipolar Rating Scales: A Survey and Novel Correlation Measures Based on Nonlinear Bipolar Scoring Functions. Information Sciences 2021, 573, 379–401. https://doi.org/10.1016/j.ins.2021.05.002.
[3] Weng, L.j. The Effects of Numerical Labels on Response Scales. Educational and Psychological Measurement 2004, 64, 956–971. https://doi.org/10.1177/0013164404268672.

T4.1	Partition-Based Rollback and Recovery
T4.2	Telemetry-Driven Recovery Triggers
T4.3	Role-Based Recovery Initiation
T4.4	Energy-Aware and Selective Retransmission
T4.5	Pre-Commit Testing and Post-Recovery Validation
T5.1	Context-Aware Update Scheduling
T5.2	Event- and Period-Based Update Triggers
T5.3	Multi-Period Update Arrangement
T5.4	Role-Based Scheduling Authorization
T5.5	Policy-Driven Scheduling Enforcement
T6.1	Standardized Packaging and Metadata
T6.2	Modular Firmware Update Strategies
T6.3	ML Model Packaging for OTA
T6.4	Storage Optimization via Compression and Caching
T6.5	Secure and Optimized Payload Formats

Table 4. Techniques and their QAs impact (QA1: Security, QA2: Scalability, QA3: Performance, QA4: Availability, QA5: Interoperability, QA6: Reliability, QA7: Privacy, QA8: Energy Management, QA9: Flexibility, QA10: Evolvability).

Mechanism	Technique	QA1	QA2	QA3	QA4	QA5	QA6	QA7	QA8	QA9	QA10
M1 Mechanism for secure and safe updates	T1.1	+2	+1	0	+1	+1	+2	0	-1	-1	0
	T1.2	+2	+1	-1	+1	+1	+1	+2	-1	+1	0
	T1.3	+2	-1	0	+1	-1	+2	+1	-1	-1	+1
	T1.4	+2	+1	-1	+1	+2	+2	+2	-1	+1	+1
	T1.5	+2	-2	-1	+1	-1	+1	-1	-2	+1	+1
	T1.6	+2	+1	-1	+1	+1	+2	0	-1	+1	+1
	T1.7	+2	+1	0	+1	+1	+2	+1	-1	+1	+1
	T1.8	+2	+1	0	+2	-1	+2	+1	-1	-1	+1
M2 Mechanism for updates management	T2.1	+1	+2	+1	+1	+2	+1	0	0	+1	+1
	T2.2	+2	+1	0	+1	+1	+1	+1	0	+1	+1
	T2.3	+1	+1	0	+1	+1	+2	0	0	+1	+1
	T2.4	+1	+2	+1	+1	+1	+1	0	+2	+2	+1
	T2.5	+1	0	0	+1	0	+1	+2	0	+1	+1

M3 Mechanism for code dissemination, propagation, and installation	T3.1	+1	+2	+1	+1	+2	+1	0	+1	+2	+1
	T3.2	+2	+1	+1	+1	+1	+2	0	0	+1	+2
	T3.3	+1	+1	+1	+2	0	+2	0	-1	0	+1
	T3.4	+1	+1	+1	+1	+1	+2	0	-1	+1	+1
	T3.5	0	+1	+2	+1	0	+1	0	+2	+1	+1
	T3.6	+2	+1	+1	+1	+1	+1	+1	-1	+1	+1
M4 Mechanism for system recovery	T4.1	+1	+1	+1	+2	0	+2	0	-1	+1	+1
	T4.2	+1	+1	0	+2	0	+2	0	-1	+1	+1
	T4.3	+2	+1	0	+1	0	+1	+1	0	+1	+1
	T4.4	0	+1	+1	+1	0	+1	0	+2	+1	+1
	T4.5	+1	+1	0	+2	0	+2	0	-1	+1	+1
M5 Mechanism for updates scheduling	T5.1	0	+1	+1	+2	0	+2	0	+2	+1	+1
	T5.2	+1	+1	0	+1	0	+1	0	0	+1	+1
	T5.3	0	+1	+1	+2	0	+1	0	+1	+2	+1
	T5.4	+2	+1	0	+1	0	+1	+1	0	+1	+1
	T5.5	+2	+1	+1	+1	0	+2	0	0	+1	+2
M6 Mechanism for elaboration and packaging	T6.1	+1	+1	+1	0	+2	+1	0	0	+1	+1
	T6.2	+1	+1	+1	+1	0	+1	0	+1	+2	+2
	T6.3	+1	+1	+1	+1	+1	+1	0	0	+1	+2
	T6.4	0	+1	+2	+1	0	+1	0	+2	0	+1
	T6.5	+2	+1	-1	+1	+1	+1	0	+1	+1	+1

Techniques Catalog: Mechanism for Secure and Safe Updates (M1)

Technique T1.1:
Firmware Integrity, Authenticity, and Installation Verification.

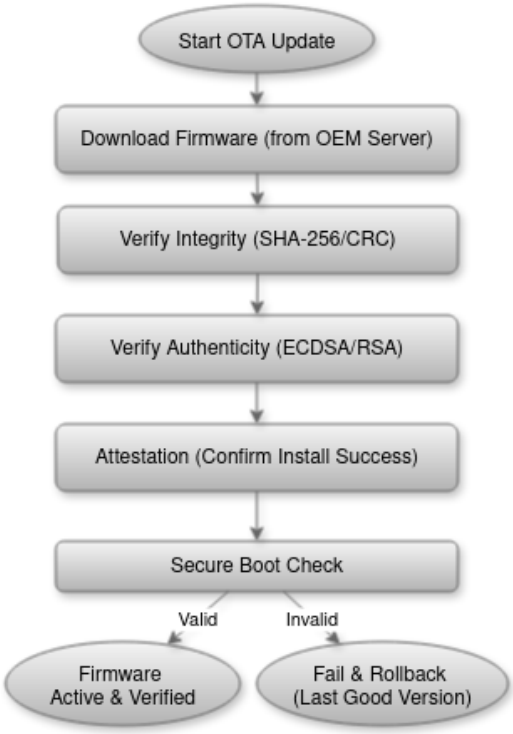
Description:

Use cryptographic hashes, such as SHA-256 or CRC, and digital signatures, such as ECDSA or RSA, to verify firmware integrity during transfer. Also, verification should be performed within trusted environments, and attestation mechanisms should be applied to confirm that the installation has been completed successfully. Secure boot mechanisms, such as ARM TrustZone, Secure Boot System Firmware Update (SBSFU), or similar solutions, are used to ensure the authenticity of firmware before it is executed.

General Use Cases:
- When devices operate on untrusted networks, the technique keeps firmware safe from tampering, whether it's being downloaded.
- When deployed in remote or unattended environments, the technique blocks the injection of unauthorized or malicious firmware.
- When multiple stakeholders are involved, the technique ensures that only authorized firmware is installed on the device.
- When devices contain sensitive data, the technique ensures firmware authenticity while reducing the risk of data leakage.
- When used on safety-critical devices, the technique prevents compromised firmware from compromising operations or safety.
- When resiliency against rollback or incomplete updates is needed, the technique relies on attestation and secure boot to confirm firmware validity.
- When communication channels' bandwidth is tight, the technique uses cryptographic hashes and digital signatures to verify the integrity of fragmented firmware transfers.
- When compliance with industry standards is a must, the technique provides proof that the firmware is authentic and meets the standards.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Hash/signatures prevent tampering.
- **Scalability**: (+1), Verification works devices population-wide.
- **Performance**: (0), Neutral.
- **Availability**: (+1), Rollback/attestation maintain uptime.
- **Interoperability**: (+1), Uses crypto standards.
- **Reliability**: (+2), Secure boot validates firmware.
- **Privacy**: (0), No direct protection of user data.
- **Energy Management**: (–1), Extra checks drain power.
- **Flexibility**: (–1), Rigid standards reduce adaptability.
- **Evolvability**: (0), Neutral.

Technique Workflow Diagram:



Technique T1.2:
Secure Payload Encryption and End-to-End Communication.

Description:

Firmware payloads should be encrypted with symmetric or asymmetric cryptographic algorithms, such as AES or RSA, and packaged in secure formats, for example COSE, JOSE, or JWS. OTA transmissions should be protected using protocols such as TLS, DTLS, or OSCORE. The use of short-lived credentials helps to maintain confidentiality, authenticity, and resilience during communication between the OEM and the device.

General Use Cases:

- When OTA transmission occurs across public or shared networks, the technique prevents data interception through encryption.

- When data is transmitted through gateways or proxies, the technique protects payloads from manipulation by untrusted nodes using encryption.

- When devices are running with limited resources, the technique lowers the chance of security breaches by implementing short-lived credentials.

- When communications occur among OEMs, servers, and devices, the technique ensures authenticity and integrity through end-to-end encryption.

- When secure communication is required, the technique prevents session hijacking and replay attacks by using TLS, DTLS, and OSCORE protocols.

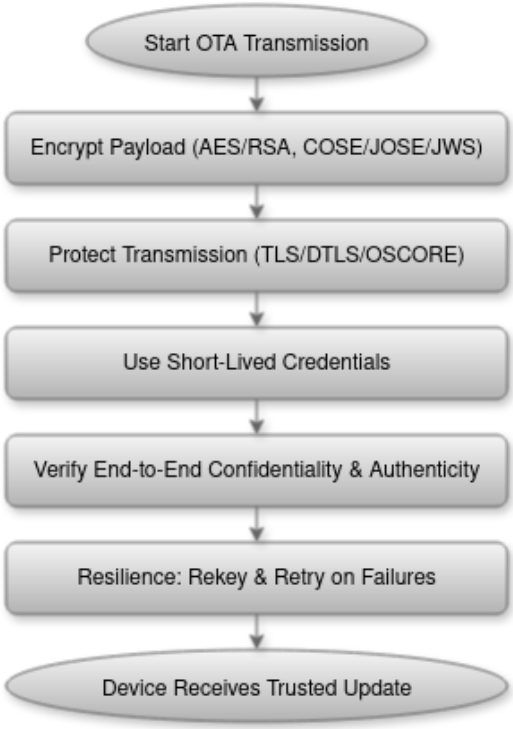
- When network connectivity is unreliable, the technique sustains security through rekeying and session retry mechanisms.

- When intellectual property must remain confidential, the technique preserves secrecy and prevents reverse engineering through encryption.

Quality Attributes (QAs) Contribution:

- **Security**: (+2), TLS/DTLS ensures confidentiality.
- **Scalability**: (+1), Applicable device population-wide.
- **Performance**: (-1), Possible crypto overhead.
- **Availability**: (+1), Prevents tampering interruptions.
- **Interoperability**: (+1), Widely adopted secure protocols.
- **Reliability**: (+1), Integrity ensured.
- **Privacy**: (+2), Protects sensitive payloads.
- **Energy Management**: (–1), Encryption drains constrained devices.
- **Flexibility**: (+1), Supports multiple protocols.
- **Evolvability**: (0), Neutral.

Technique Workflow Diagram:



Technique T1.3:
Trusted Execution and Runtime Protection.

Description:

Safeguard the integrity and confidentiality of the OTA update process during installation and execution by isolating sensitive operations within secure, trusted, or hardened execution environments. Secure environments include Trusted Execution Environments (TEEs), hypervisors, or hardened operating system contexts, which specifically protect cryptographic keys, update logic, and system state. By isolating these components, sensitive processes and data are shielded from compromise throughout both installation and execution phases, particularly important for devices exposed to physical access or hostile runtime conditions.

General Use Cases:

- When sensitive cryptographic keys are used, the technique ensures safety with TEEs, preventing unauthorized access and stopping data from leaking.

- When updating critical system components, the technique reduces the risk of interference or compromise by isolating update processes through hypervisors and operating system layers.

- When operating in untrusted or adversarial environments, the technique maintains update security through TEEs, even if the operating system is compromised.

- When authenticated execution is required, the technique ensures only verified code runs by enabling secure boot and attestation through TEEs.

- When devices operate in multi-tenant environments, the technique prevents cross-tenant interference by using hypervisors to isolate update processes from other workloads.

- When integrity requirements need to be enforced during runtime, the technique guarantees the integrity of critical components through TEEs and OS-level protections.

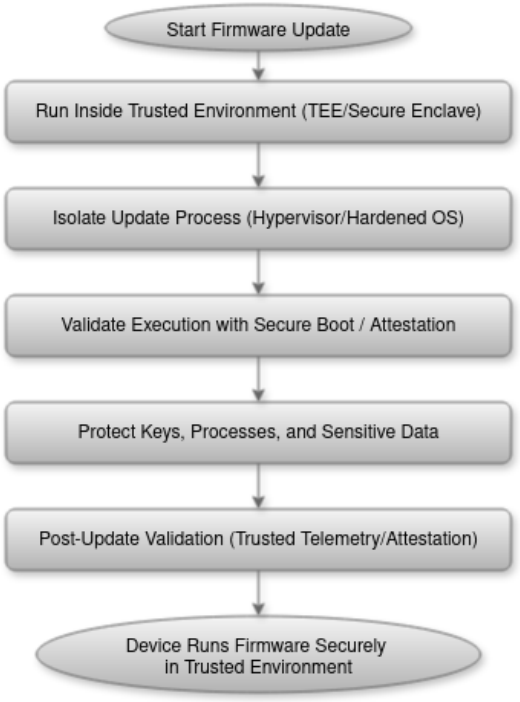
- When proprietary code and data require protection, the technique prevents reverse engineering by relying on encrypted execution environments.

- When performing post-update validation, the technique ensures the confidentiality and integrity of telemetry and attestation data through runtime protections.

Quality Attributes (QAs) Contribution:

- **Security**: (+2), TEEs isolate processes/keys.
- **Scalability**: (–1), TEEs not on all devices.
- **Performance**: (0), Neutral.
- **Availability**: (+1), Isolation reduces runtime failures.
- **Interoperability**: (–1), Vendor-specific implementations.
- **Reliability**: (+2), Stable secure execution.
- **Privacy**: (+1), Runtime data protected.
- Energy Management: (–1), Runtime overhead.
- **Flexibility**: (–1), Hardware-limited.
- **Evolvability**: (+1), Trusted runtime supports adaptability.

Technique Workflow Diagram:



Technique T1.4:
Key, Credential, and Identity Management.

Description:

Generate, provision, and manage cryptographic keys and digital certificates to facilitate secure device-server authentication. System must maintain unique device identities, and robust authentication credentials, such as multi-factor authentication or SIM-based methods, should be pre-installed and configured. Additionally, the System must enforce role-based access control.

General Use Cases:

- When devices require cloud authentication, the technique establishes secure connections by employing digital certificates and cryptographic keys.

- When software updates are delivered, the technique restricts access exclusively to authorized devices by using unique device identifiers and credentials, preventing unauthorized or rogue device access.

- When update authorization is needed, the technique enforces separation of duties through role-based OTA control, ensuring that only approved personnel can initiate software updates.

- When long-term trust must be preserved, the technique maintains security by periodically rotating, renewing, or revoking cryptographic keys and digital certificates.

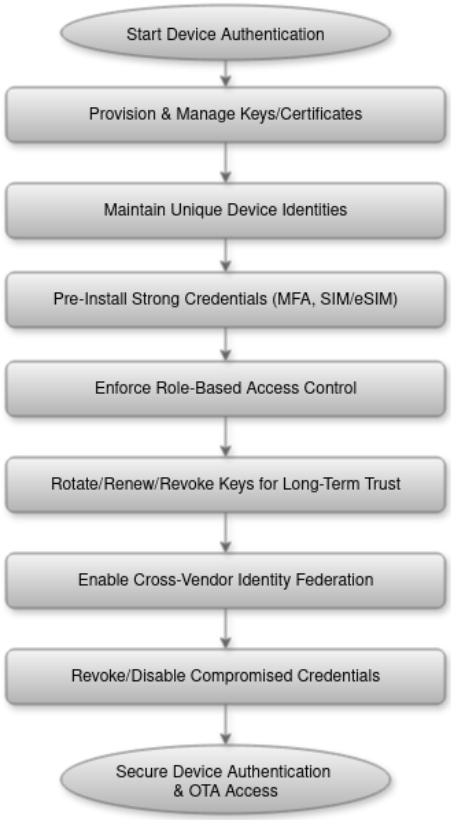
- When identity validation is required, the technique prevents spoofing through strong authentication mechanisms such as MFA, security tokens, or SIM-based methods.

- When devices operate in multi-tenant ecosystems, the technique ensures interoperability and system federation by assigning unique device identifiers across organizational domains.

- When devices become compromised, the technique isolates them by revoking credentials, effectively blocking their access to OTA updates and network resources.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Strong auth blocks unauthorized access.
- **Scalability**: (+1), Works across device populations.
- **Performance**: (-1), Minor auth delay.
- **Availability**: (+1), Secure connections maintain service.
- **Interoperability**: (+2), Based on standards.
- **Reliability**: (+2), Prevents spoofing/rogue devices.
- **Privacy**: (+2), Credentials protect identity/data.
- **Energy Management**: (–1), Auth consumes resources.
- **Flexibility**: (+1), Supports rotation/revocation.
- **Evolvability**: (+1), PKI/MFA adapt over time.

Technique Workflow Diagram:



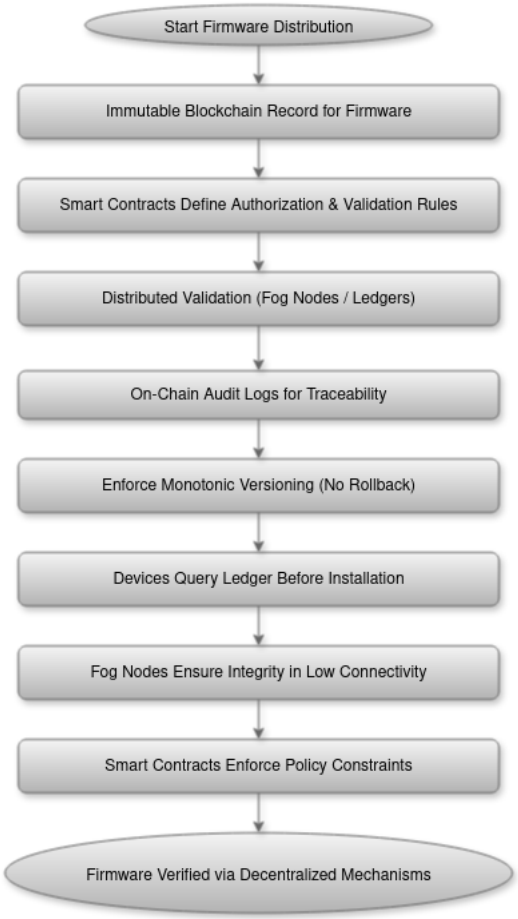
Technique T1.5:
Decentralized and Blockchain-Based Verification.

Description:
Use blockchain, smart contracts, fog nodes, or similar decentralized methods to verify firmware authenticity, track delivery, enforce constraints, and allow devices to perform transaction-based validation queries before installation.

General Use Cases:
- When tamper-proof distribution is required, the technique provides proof that updates remain unchanged by recording them on a blockchain.
- When multiple stakeholders participate in the update process, the technique defines authorization and validation rules through smart contracts.
- When operating in decentralized IoT ecosystems, the technique ensures firmware authenticity by enabling devices to validate updates via blockchain ledgers or fog nodes.
- When transparent auditability is needed, the technique enables traceability and compliance using on-chain logs.
- When updates must resist rollback or replay attacks, the technique enforces monotonic versioning through blockchain mechanisms.
- When device-driven validation is required, the technique confirms update authenticity by allowing devices to query the blockchain ledger.
- When devices operate in partially connected networks, the technique ensures integrity by leveraging fog-based validators without requiring full cloud access.
- When automatic policy enforcement is necessary, the technique checks the device type or compliance requirements before updates are applied using smart contracts.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Blockchain ensures tamper-proof integrity.
- **Scalability**: (–2), Blockchain overhead limits device population scale.
- **Performance**: (–2), Slow ledger validation.
- **Availability**: (+1), Distributed validation aids resilience.
- **Interoperability**: (–1), Varying blockchain frameworks.
- **Reliability**: (+1), Consistency enforced.
- **Privacy**: (–1), Ledger exposes data.
- **Energy Management**: (–2), Consensus is energy-intensive.
- **Flexibility**: (+1), Smart contracts adaptable.
- **Evolvability**: (+1), Ledgers/logic upgradable.

Technique Workflow Diagram:



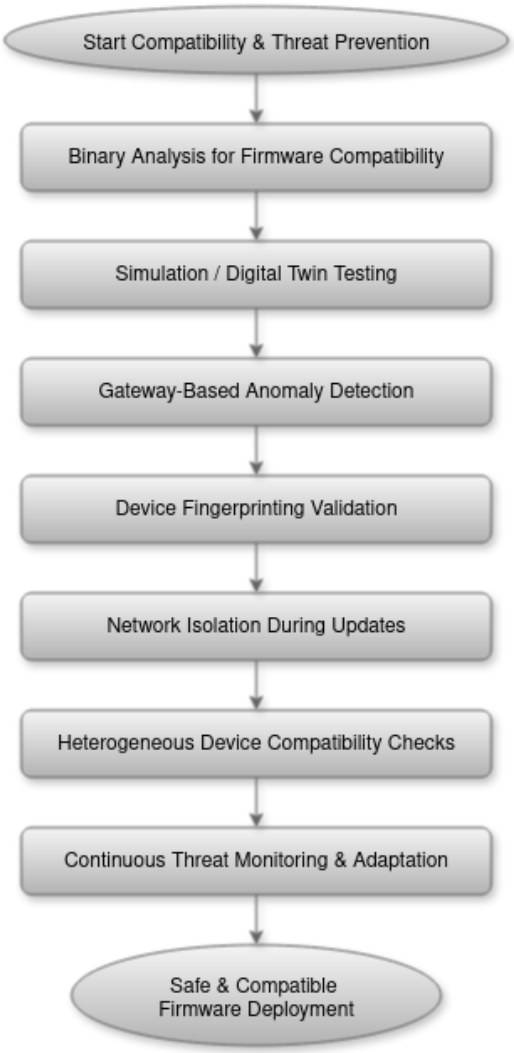
Technique T1.6:
Compatibility and Threat Prevention.

Description:
Perform compatibility checks via binary analysis or simulation/digital twins before deployment. Detect and prevent threats using gateway-based anomaly detection, device fingerprinting, and network isolation to block unauthorized updates.

General Use Cases:
- When pre-deployment compatibility must be verified, the technique confirms hardware–software matching through binary analysis and simulation.
- When advanced behavior validation is required, the technique detects potential failures before rollout by using digital twins or sandbox environments.
- When defending against unauthorized updates, the technique blocks suspicious traffic through anomaly-detection gateways.
- When device-specific validation is needed, the technique ensures updates target the correct hardware by applying device fingerprinting.
- When secure delivery environments are necessary, the technique blocks intruders and fake updates through network isolation.
- When operating in heterogeneous ecosystems, the technique reduces cross-device failures by performing compatibility checks.
- When resilience to new threats is required, the technique detects emerging attack patterns through continuous monitoring.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Anomaly detection blocks threats.
- **Scalability**: (+1), Checks apply device population-wide.
- **Performance**: (-1), Analysis overhead manageable.
- **Availability**: (+1), Prevents faulty deployment.
- **Interoperability**: (+1), Simulations ensure compatibility.
- **Reliability**: (+2), Sandbox/digital twins prevent failure.
- **Privacy**: (0), No direct protection of personal data.
- **Energy Management**: (–1), Sandbox/analysis consumes power.
- **Flexibility**: (+1), Adjustable to device types.
- **Evolvability**: (+1), Models adapt to new threats.

Technique Workflow Diagram:



Technique T1.7:
Protocol, Debug Interface, and Request Security.

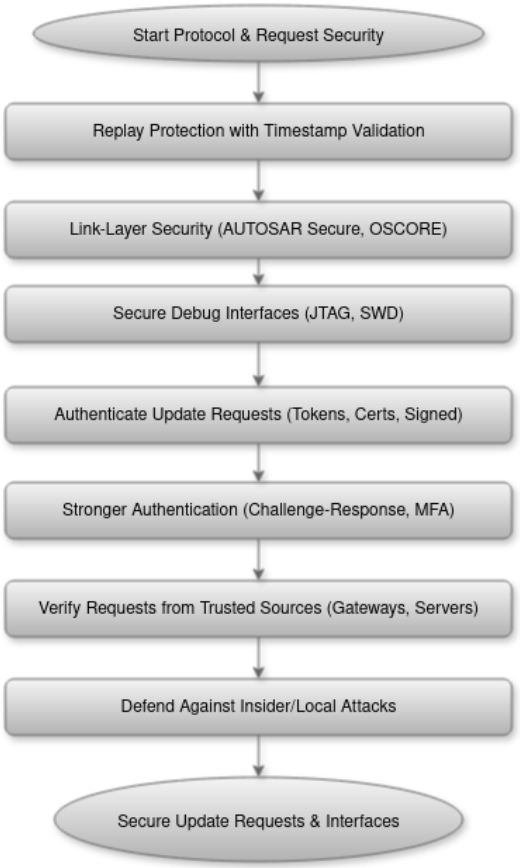
Description:
Prevent unauthorized update requests, protocol-level attacks, and exploitation of debug interfaces by securing communication protocols and restricting low-level system access using timestamp validation, integrating link-layer security (e.g., AUTOSAR Secure Onboard Communication (SecOC), securing debug interfaces (e.g., JTAG, SWD), and authenticating update requests using tokens, challenge-response, or Multi-Factor Authentication (MFA). This technique addresses threats such as replay attacks, unauthorized firmware injection, and misuse of debug ports, ensuring that OTA update requests follow authenticated and validated communication paths.

General Use Cases:
- When defending against replay attacks, the technique ensures update freshness through the validation of timestamps.
- When link-layer security is required, the technique protects OTA data integrity by using AUTOSAR Secure or OSCORE.
- When securing debug interfaces, the technique prevents bypass attacks by locking or disabling JTAG/SWD access.
- When authenticating update requests, the technique validates the legitimacy of updates using tokens or digital certificates.
- When protecting sensitive devices, the technique prevents unauthorized updates through challenge-response mechanisms or MFA.
- When operating in distributed environments, the technique confirms that update requests originate from trusted sources through verification processes.
- When defending against local attacks, the technique blocks malicious access through secure debug controls combined with strong authentication.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Tokens/replay protection block attacks.
- **Scalability**: (+1), Protocols apply device population-wide.
- **Performance**: (0), Neutral.
- **Availability**: (+1), Prevents denial of update access.
- **Interoperability**: (+1), Common protocol support.

- **Reliability**: (+2), Valid requests ensure safe updates.
- **Privacy**: (+1), Auth improves confidentiality.
- **Energy Management**: (–1), Crypto drains resources.
- **Flexibility**: (+1), Supports multiple auth methods.
- **Evolvability**: (+1), Auth standards are upgradeable.

Technique Workflow Diagram:



Technique T1.8:
Installation and Runtime Security Assurance.

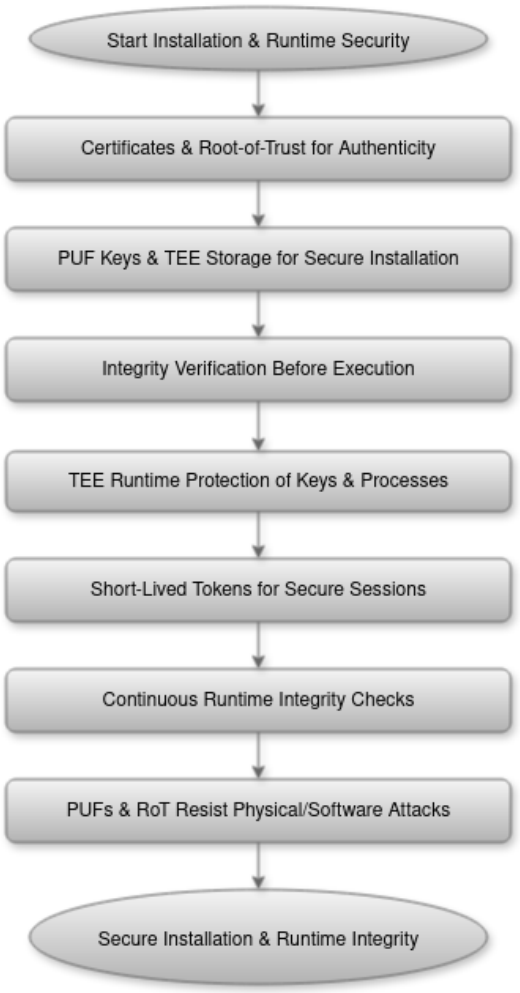
Description:
Maintain long-term firmware integrity and system trust by enforcing secure installation conditions and continuous runtime verification using certificates, Root of Trust, Physically Unclonable Functions (PUFs), TEE storage, and short-lived tokens to secure installation processes and maintain firmware integrity during runtime. This technique establishes a persistent root of trust that ensures only authenticated and validated firmware remains active across reboots and update cycles, supporting sustained system reliability and resistance to post-installation tampering.

General Use Cases:
- When software is installed for the first time, the technique verifies authenticity through certificates and a Root of Trust (RoT).
- When the update process is underway, the technique prevents tampering by protecting keys and data with PUF-derived secrets and TEE-based storage.
- When firmware is about to execute, the technique ensures code integrity by confirming it matches the signed version.
- When runtime protection is required, the technique secures cryptographic keys and sensitive operations through TEEs.
- When using short-lived credentials, the technique prevents session reuse or hijacking by utilizing time-bound tokens.
- When validating post-update integrity, the technique blocks unauthorized modifications through runtime checks.
- When facing physical or software-based attacks, the technique protects secrets and firmware through PUFs and a Root of Trust.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Root-of-Trust/PUF ensure integrity.
- **Scalability**: (+1), Applied in secure-capable devices.
- **Performance**: (0), Runtime checks acceptable.
- **Availability**: (+2), Prevents invalid firmware execution.
- **Interoperability**: (–1), Hardware-specific features.
- **Reliability**: (+2), Continuous validation protects runtime.

- **Privacy**: (+1), Protects sensitive runtime operations.
- **Energy Management**: (–1), Runtime verification consumes resources.
- **Flexibility**: (–1), Tied to hardware capabilities.
- **Evolvability**: (+1), Secure baseline supports future updates.

Technique Workflow Diagram:



Techniques Catalog: Mechanism for Updates Management (M2)

Technique T2.1:
OTA Orchestration and Management Interfaces.

Description:
Use centralized or distributed OTA platforms (e.g., Hawkbit, ZT-OTA), web/mobile dashboards, and lifecycle APIs (RESTful CRUD, polling, event triggers) to manage firmware updates, deployment, and logging.

General Use Cases:
- When managing extensive device populations, the technique enables bulk updates and centralized or distributed monitoring through orchestration platforms.
- When visibility and operational control are required, the technique provides tracking, error reporting, and progress insights through dashboards and APIs, enabling seamless monitoring and management.
- When deployments operate across regions or device vendors, the technique manages heterogeneous devices through orchestration platforms.
- When workflow consistency is necessary, the technique enforces approval, deployment, and rollback via lifecycle management APIs.
- When integrated with CI/CD pipelines, the technique pushes updates automatically by triggering OTA deployments upon new firmware builds.
- When compliance is required, the technique documents update history and standards conformance through logs and audit reports.
- When downtime is a concern, the technique schedules updates flexibly through operator interfaces supporting timed and staged deployments.
- When serving multiple independent device groups, the technique ensures secure update management by applying role-based access control on orchestration platforms.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Dashboards/APIs enforce access control.
- **Scalability**: (+2), Centralized/distributed platforms manage device populations.
- **Performance**: (+1), Orchestration optimizes scheduling.
- **Availability**: (+1), Lifecycle APIs support rollback/recovery.
- **Interoperability**: (+2), Standardized interfaces ease integration.
- **Reliability**: (+1), Structured management reduces errors.
- **Privacy**: (0), No direct impact on user data.
- **Energy Management**: (0), Orchestration not energy-related.
- **Flexibility**: (+1), Operators configure timing and policies.
- **Evolvability**: (+1), APIs adapt to new workflows.

Technique Workflow Diagram:

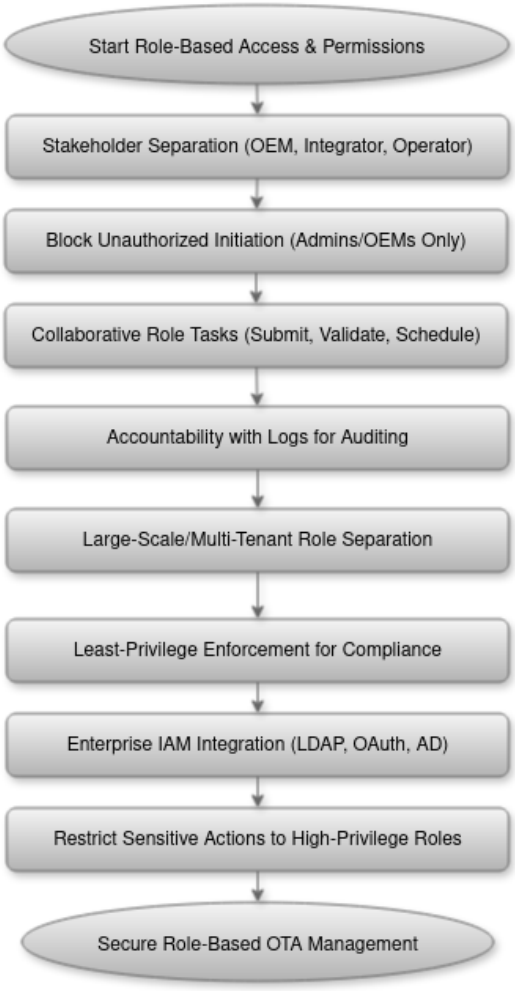
Technique T2.2:
Role-Based Access and Permissions.

Description:
Define update actors (e.g., OEM, integrator, uploader, user) and assign role-based permissions to control who can initiate, approve, or manage firmware updates.

General Use Cases:
- When stakeholder separation is required, the technique assigns distinct update permissions to OEMs, integrators, and operators.
- When preventing unauthorized initiation is necessary, the technique ensures only approved roles (e.g., admin or OEM) can trigger updates.
- When collaborative management workflows are used, the technique divides responsibilities by assigning upload, validation, and scheduling tasks to different roles.
- When accountability is essential, the technique enables auditability by tracking actions and approvals through logs.
- When operating at a large scale or across multiple independent device groups, the technique separates permissions by group or customer using role-based access control.
- When compliance requirements must be met, the technique supports certifications and standards by enforcing least-privilege role separation.
- When integrating with enterprise environments, the technique connects role enforcement to IAM systems such as LDAP, OAuth, or Active Directory.
- When performing sensitive operations, the technique restricts critical actions to high-privilege roles to prevent misuse or escalation.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Restricts update actions to authorized roles.
- **Scalability**: (+1), Roles manage large setups with separation of concerns.
- **Performance**: (0), Role checks have negligible effect.
- **Availability**: (+1), Controlled access avoids system abuse.
- **Interoperability**: (+1), Integrates with IAM/LDAP/OAuth.
- **Reliability**: (+1), Prevents accidental/malicious initiations.
- **Privacy**: (+1), Limits exposure of operations by role.
- **Energy Management**: (0), No relation.
- **Flexibility**: (+1), Supports multi-tenant permissions.
- **Evolvability**: (+1), Roles adapt to organizational changes.

Technique Workflow Diagram:



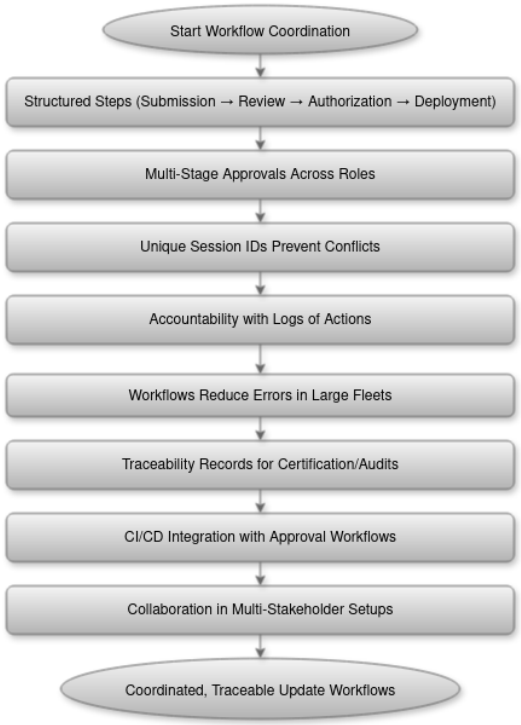
Technique T2.3:
Workflow Coordination and Traceability.

Description:
Manage the update process through structured workflows (submission, review, authorization, deployment) with unique session identifiers for tracking and conflict prevention.

General Use Cases:
- When structured lifecycles are required, the technique ensures consistency by enforcing defined update steps such as submission, review, authorization, and deployment.
- When multi-stage approvals are necessary, the technique distributes validation across roles (integrator, OEM, operator) to ensure proper governance and control.
- When preventing operational conflicts is critical, the technique stops overlapping update sessions by enforcing session IDs.
- When accountability is needed, the technique tracks responsibility by logging who approved or deployed updates.
- When operating across large device populations, the technique reduces errors during simultaneous or staged rollouts through orchestrated workflows.
- When certification or audits are required, the technique provides traceability records that prove compliance with update policies.
- When integrated with CI/CD pipelines, the technique moves updates from build systems into formal approval workflows to enforce governance and ensure updates are approved.
- When multiple stakeholders participate, the technique ensures collaboration without bypassing security checks by enforcing controlled workflows.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Approvals/traceability prevent malicious bypass.
- **Scalability**: (+1), Structured rollouts handle device populations.
- **Performance**: (0), Overhead minimal.
- **Availability**: (+1), Coordinated deployments minimize downtime.
- **Interoperability**: (+1), Can integrate across tools/platforms.
- **Reliability**: (+2), Structured workflows reduce failures.
- **Privacy**: (0), Workflows don’t affect data confidentiality.
- **Energy Management**: (0), No direct impact.
- **Flexibility**: (+1), Multi-role collaboration supports varied orgs.
- **Evolvability**: (+1), Workflows extend with CI/CD integration.

Technique Workflow Diagram:



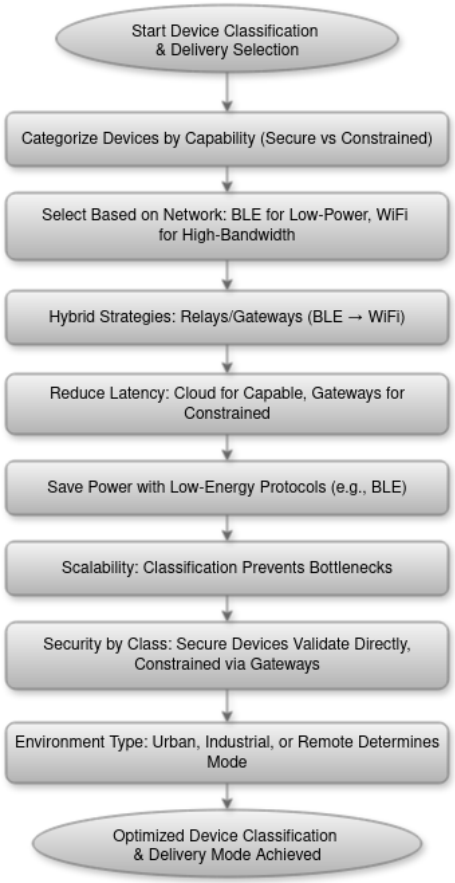
Technique T2.4:
Device Classification and Delivery Mode Selection.

Description:
Categorize devices by capability (secure-capable vs constrained) and choose appropriate delivery methods (WiFi direct, BLE to WiFi, BLE-only) to optimize update efficiency.

General Use Cases:
- When managing heterogeneous device populations, the technique ensures optimal delivery selection by categorizing devices based on their capabilities.
- When operating under network constraints, the technique enhances efficiency by utilizing BLE for low-power scenarios and Wi-Fi for high-bandwidth requirements.
- When hybrid delivery strategies are required, the technique supports constrained devices by routing updates through gateways or relays (e.g., BLE, Wi-Fi).
- When reducing latency is critical, the technique enables capable devices to update directly from the cloud, while constrained devices update via gateways or peer-to-peer (P2P) connections.
- When minimizing power consumption is necessary, the technique reduces battery drain by leveraging low-energy protocols such as BLE.
- When scaling to large device populations, the technique prevents bottlenecks by classifying devices and assigning update paths accordingly.
- When applying security based on device class, the technique ensures secure devices validate updates directly, while constrained devices rely on gateways.
- When adapting to deployment environments, the technique selects update modes based on whether devices operate in urban, industrial, or remote settings.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Gateways validate constrained devices.
- **Scalability**: (+2), Avoids bottlenecks via classification.
- **Performance**: (+1), Delivery optimization improves efficiency.
- **Availability**: (+1), Hybrid modes reduce downtime.
- **Interoperability**: (+1), Delivery adapts to protocol diversity.
- **Reliability**: (+1), Ensures devices get correct delivery method.
- **Privacy**: (0), Classification doesn’t affect user data.
- **Energy Management**: (+2), Low-energy protocols save battery.
- **Flexibility**: (+2), Adaptive delivery per device type.
- **Evolvability**: (+1), Supports heterogeneous and future devices.

Technique Workflow Diagram:



Technique T2.5:
User Awareness and Consent Management.

Description:
Notify users before updates, allow them to approve or postpone installations, and provide clear summaries of changes before and after the update.

General Use Cases:
- When user control is required, the technique gives users full authority to decide update timing, ensuring they maintain control over their experience.
- When avoiding disruption is important, the technique minimizes workflow interruptions by allowing users to schedule updates at convenient times.
- When transparency is required, the technique enhances trust by providing pre- and post-update summaries to address the unique needs of each case.
- When devices are in critical operation, the technique prevents updates by requiring user consent during sensitive tasks.
- When regulatory compliance is applicable, the technique ensures explicit user awareness and approval, thereby meeting legal requirements.
- When devices are shared, the technique prevents unilateral updates by requiring consent from all users.
- When user acceptance must be maximized, the technique offers clear explanations to increase trust.
- When optional features are offered, the technique enables tailored functionality by providing staged or selectable upgrade options.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Approval prevents unauthorized updates.
- **Scalability**: (0), Individual approvals don't scale device populations.
- **Performance**: (0), Notifications have negligible cost.
- **Availability**: (+1), Scheduling by user reduces disruption.
- **Interoperability**: (0), Mostly UI/consent flow, not integration.
- **Reliability**: (+1), User control avoids untimely failures.
- **Privacy**: (+2), User consent protects personal data use.
- **Energy Management**: (0), Not energy-related.
- **Flexibility**: (+1), Staged/optional features configurable.
- **Evolvability**: (+1), Adapts to new compliance requirements.

Technique Workflow Diagram:

Techniques Catalog: Mechanism for Code Dissemination, Propagation, and Installation (M3)

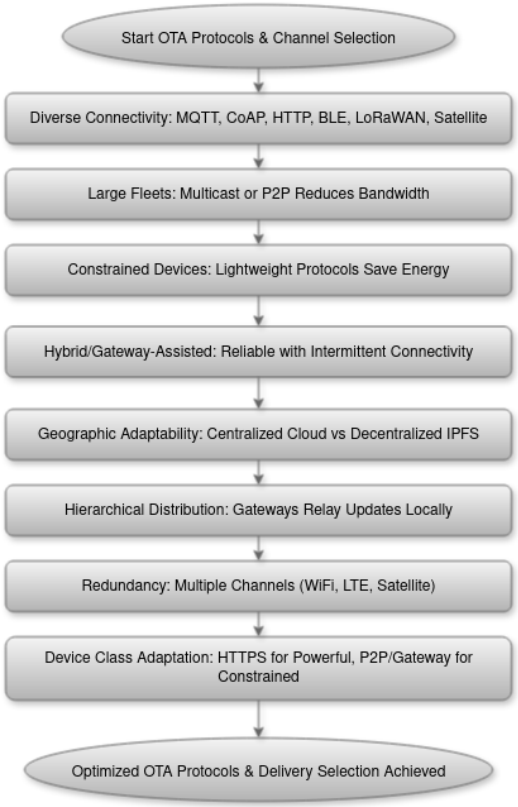
Technique T3.1:
OTA Protocols, Delivery Models, and Channel Selection.

Description:
Disseminate updates using MQTT, CoAP, HTTP(S), BLE mesh, LoRaWAN multicast, satellite, wired, or hybrid push-pull approaches. Select centralized, gateway, peer-to-peer, or decentralized (e.g., IPFS) models based on scalability, reach, and device capabilities.

General Use Cases:
- When diverse connectivity environments exist, the technique supports Wi-Fi, BLE, LoRaWAN, and satellite by using flexible protocols such as MQTT, CoAP, and HTTP.
- When managing large device populations, the technique reduces bandwidth consumption during mass updates through multicast or peer-to-peer (P2P) delivery.
- When operating with constrained devices, the technique minimizes energy and data usage by employing lightweight communication protocols.
- When connectivity is intermittent, the technique ensures reliable delivery through hybrid or gateway-assisted update mechanisms.
- When geographic adaptability is required, the technique selects centralized cloud servers for connected regions.
- When hierarchical distribution is beneficial, the technique relays updates from the cloud to local devices through gateways.
- When redundancy is required, the technique maintains reliability by utilizing multiple communication channels, such as Wi-Fi, LTE, and satellite.
- When devices differ in capability, the technique assigns HTTPS to powerful nodes while constrained devices rely on P2P or gateway-mediated updates.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Secure protocols like HTTPS/DTLS.
- **Scalability**: (+2), Multicast/P2P reduce device population load.
- **Performance**: (+1), Optimized channel selection improves speed.
- **Availability**: (+1), Hybrid/gateway models maintain service.
- **Interoperability**: (+2), Broad protocol support (MQTT, CoAP, LoRa, etc.).
- **Reliability**: (+1), Redundancy across channels improves delivery.
- **Privacy**: (0), Protocols alone don’t protect data.
- **Energy Management**: (+1), Lightweight protocols conserve energy.
- **Flexibility**: (+2), Adaptive by network/device type.
- **Evolvability**: (+1), Hybrid models extend future adaptability.

Technique Workflow Diagram:



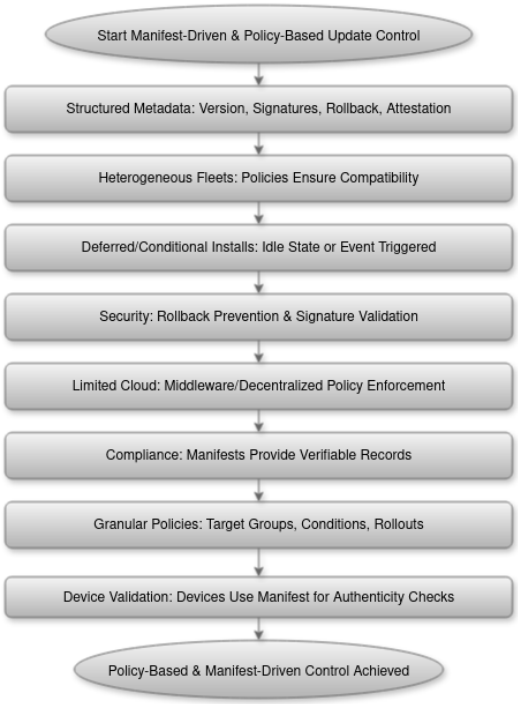
Technique T3.2:
Manifest-Driven and Policy-Based Update Control.

Description:
Use dedicated channels for manifest distribution, including metadata, signatures, version rules, rollback constraints, and attestation data. Support deferred or event-triggered installation and coordinate updates with middleware or decentralized logic enforcing policies across devices.

General Use Cases:
- When structured metadata is required, the technique provides versioning, signatures, rollback rules, and attestation through update manifests.
- When managing heterogeneous device populations, the technique ensures that only compatible devices receive updates by enforcing policy rules.
- When installs must be deferred or triggered conditionally, the technique postpones updates until devices are idle or specific events occur.
- When security constraints are in place, the technique prevents malicious updates through rollback protection and signature verification.
- When cloud access is limited, the technique enforces update policies locally through middleware or decentralized logic.
- When compliance must be demonstrated, the technique provides verifiable delivery and installation records via manifests.
- When granular policy control is required, the technique enables administrators to target specific device groups, conditions, or staggered rollout schedules.
- When devices perform self-validation, the technique ensures authenticity by enabling devices to verify manifest data before installation.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Signatures, rollback rules, attestation.
- **Scalability**: (+1), Manifests distribute policies device population-wide.
- **Performance**: (+1), Structured updates optimize rollout.
- **Availability**: (+1), Controlled installs reduce outages.
- **Interoperability**: (+1), Standardized manifests (RFC, JSON).
- **Reliability**: (+2), Manifests enforce consistency.
- **Privacy**: (0), Manifests don’t handle personal data.
- **Energy Management**: (0), Not energy-related.
- **Flexibility**: (+1), Deferred/event-driven installation.
- **Evolvability**: (+2), Policies adapt to future conditions.

Technique Workflow Diagram:



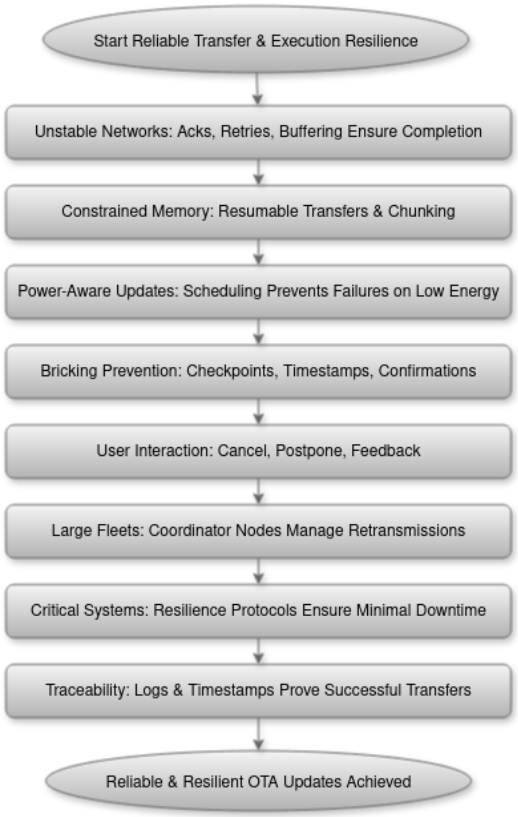
Technique T3.3:
Reliable Transfer and Execution Resilience.

Description:
Ensure that OTA updates are delivered and applied correctly despite network instability, device restarts, or power interruptions. This technique incorporates operations such as acknowledgments, buffering, retries, resumable transfers, and execution checkpoints to prevent partial or corrupted updates. It is especially relevant in large-scale or unreliable network environments, where resilience is required to preserve system availability and prevent inconsistent device states.

General Use Cases:
- When networks are unstable, the technique ensures update completion through acknowledgments, retries, and buffering mechanisms.
- When devices have constrained memory, the technique handles large updates by using resumable transfers and chunked delivery.
- When power availability is limited, the technique avoids update failures by scheduling updates based on energy conditions.
- When preventing device bricking is critical, the technique verifies full delivery through checkpointing, timestamps, and confirmation steps.
- When updating large device populations, the technique reduces congestion and manages retransmissions through the use of coordinator nodes.
- When maintaining uptime is essential for critical systems, the technique ensures minimal downtime through resilience protocols.
- When traceability is required, the technique proves successful transfers by recording logs and timestamps.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Timestamps and confirmations prevent replay.
- **Scalability**: (+1), Coordinator nodes manage large device populations.
- **Performance**: (+1), Optimized retransmission improves efficiency.
- **Availability**: (+2), Prevents bricking with checkpointing.
- **Interoperability**: (0), Basic transfer, not format-driven.
- **Reliability**: (+2), Retries, buffering, acknowledgments.
- **Privacy**: (0), No direct effect.
- **Energy Management**: (–1), Retries and checkpoints consume extra power.
- **Flexibility**: (0), Process-focused, not adaptation-driven.
- **Evolvability**: (+1), Resilience protocols adapt to new conditions.

Technique Workflow Diagram:



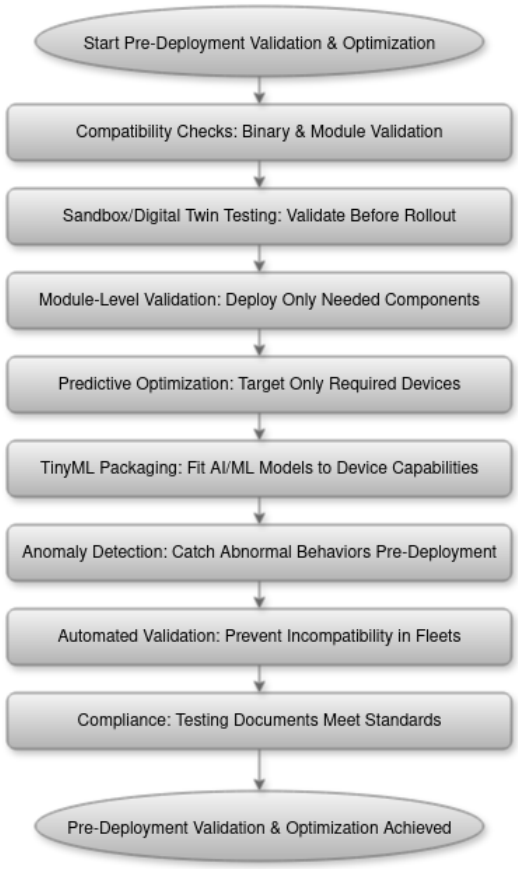
Technique T3.4:
Pre-Deployment Validation and Optimization.

Description:
Perform compatibility checks, sandbox testing, and module validation before deployment. Enhance targeting and performance using TinyML-based deployment, model-specific packaging, anomaly detection, and predictive optimization.

General Use Cases:
- When compatibility must be verified, the technique confirms hardware and OS alignment through binary and module validation.
- When minimizing live failures is critical, the technique validates updates in sandbox or digital twin environments before rollout.
- When devices are constrained, the technique deploys only the required components by using module-level validation.
- When updates must be targeted, the technique ensures only necessary devices receive updates through predictive optimization.
- When using AI/ML models on edge devices, the technique fits model packages to device capabilities through TinyML packaging.
- When anomalies must be detected early, the technique identifies abnormal behaviors or dependencies through pre-update checks.
- When scaling to large device populations, the technique prevents incompatible deployments through automated validation workflows.
- When compliance must be demonstrated, the technique documents validation results to prove adherence to required standards.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Anomaly detection spots threats pre-deployment.
- **Scalability**: (+1), Validation automates across device populations.
- **Performance**: (+1), Predictive optimization reduces load.
- **Availability**: (+1), Reduces live downtime failures.
- **Interoperability**: (+1), Checks ensure hardware/OS match.
- **Reliability**: (+2), Sandbox/digital twin testing prevents failure.
- **Privacy**: (0), No personal data protection.
- **Energy Management**: (–1), Sandbox/validation consumes resources.
- **Flexibility**: (+1), Supports modular/targeted updates.
- **Evolvability**: (+1), ML-based optimization adapts over time.

Technique Workflow Diagram:



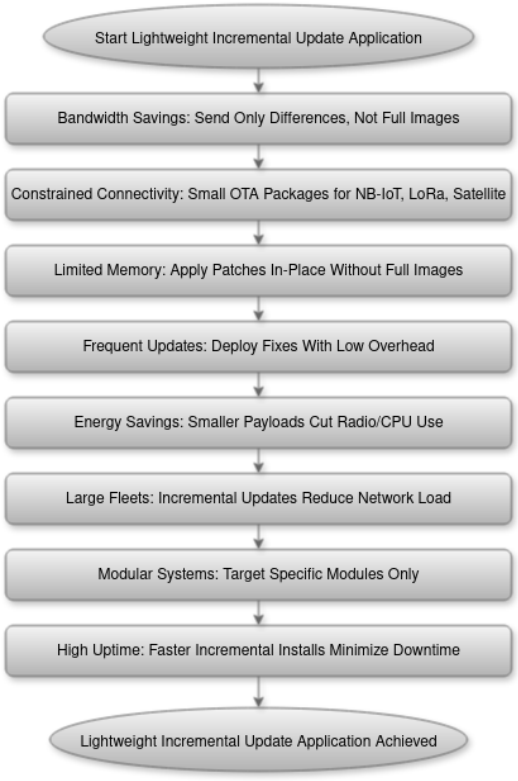
Technique T3.5:
Lightweight Incremental Update Application.

Description:
Reduce update size, transmission overhead, and installation time by applying firmware or software updates incrementally rather than replacing full images. This technique updates only the modified portions of the system through insert, modify, delete, or copy operations in memory, making it particularly suitable for bandwidth-constrained, energy-limited, or intermittently connected IoT devices. Lightweight incremental updates improve availability and energy efficiency by minimizing update duration and reducing disruption during the update process.

General Use Cases:
- When bandwidth savings are required, the technique reduces data usage by transmitting only incremental differences, rather than full images.
- When connectivity is constrained, the technique enables OTA updates on NB-IoT, LoRa, and satellite devices by delivering small update packages.
- When device memory is limited, the technique applies patches in-place without requiring the storage of full firmware images.
- When updates occur frequently, the technique deploys lightweight incremental patches to deliver fixes with minimal overhead.
- When conserving energy is important, the technique extends battery life by reducing radio and CPU usage through smaller payloads.
- When scaling across large device populations, the technique improves rollout reliability by reducing network load through incremental updates.
- When systems are modular, the technique lowers update risk and complexity by targeting specific components instead of replacing the entire image.
- When high uptime is required, the technique minimizes downtime by enabling faster installation of incremental updates.

Quality Attributes (QAs) Contribution:
- **Security**: (0), No direct protection unless combined.
- **Scalability**: (+1), Incremental updates device population-wide.
- **Performance**: (+2), Smaller payloads improve transfer speed.
- **Availability**: (+1), Fast installs reduce downtime.
- **Interoperability**: (0), Patch format varies, not universal.
- **Reliability**: (+1), Modular rollback reduces total failures.
- **Privacy**: (0), No direct impact.
- **Energy Management**: (+2), Reduced CPU/radio saves power.
- **Flexibility**: (+1), Modular patching supports adaptability.
- **Evolvability**: (+1), Incremental fixes extend lifecycle.

Technique Workflow Diagram:



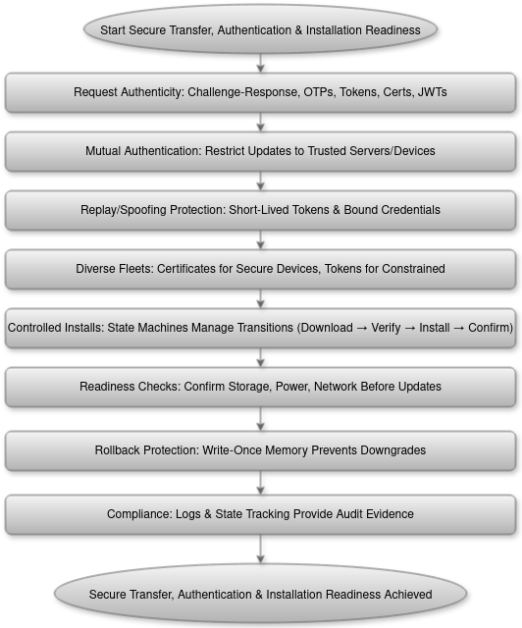
Technique T3.6:
Secure Transfer, Authentication, and Installation Readiness.

Description:
Protect the OTA dissemination and installation process against unauthorized access, replay attacks, and unsafe installation conditions. Use methods such as challenge-response, One-Time Password (OTP), bound tokens, certificates, mutual authentication, or JSON Web Token (JWT). Monitor system readiness and manage update state transitions using state machines. Prevent rollback by employing write-once memory. These measures directly reduce risks such as unauthorized modification, downgrading critical software, and installing updates on unprepared devices. By combining authentication methods, update state management, and rollback prevention, the likelihood of malicious updates, software version inconsistencies, and system instability during dissemination is reduced.

General Use Cases:
- When request authenticity must be verified, the technique authenticates devices and servers using challenge-response, OTPs, tokens, certificates, or JWTs.
- When preventing unauthorized initiation is required, the technique restricts updates to trusted endpoints through mutual authentication.
- When defending against replay or spoofing attacks, the technique prevents request reuse by employing short-lived tokens and bound credentials.
- When supporting diverse device populations, the technique ensures coverage by using flexible authentication methods, such as certificates for secure devices and tokens for constrained ones.
- When controlled installation sequencing is necessary, the technique enforces update flow through state machines that manage download, verification, installation, and confirmation steps.
- When readiness must be validated, the technique checks device storage, power, and network availability before initiating updates.
- When rollback protection is required, the technique prevents downgrades by locking firmware versions in write-once memory.
- When compliance must be demonstrated, the technique provides verifiable audit evidence through logs and update state tracking.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Challenge-response, certificates, and tokens.
- **Scalability**: (+1), Flexible authentication across device populations.
- **Performance**: (+1), Streamlined readiness avoids wasted attempts.
- **Availability**: (+1), Readiness checks reduce failed installs.
- **Interoperability**: (+1), Supports constrained and secure devices.
- **Reliability**: (+1), State machine ensures safe transitions.
- **Privacy**: (+1), JWT/certificates protect data exchanges.
- **Energy Management**: (–1), Crypto/auth overhead.
- **Flexibility**: (+1), Configurable enforcement by context.
- **Evolvability**: (+1), Flexible authentication adapts to device classes.

Technique Workflow Diagram:



Techniques Catalog: Mechanism for System Recovery (M4)

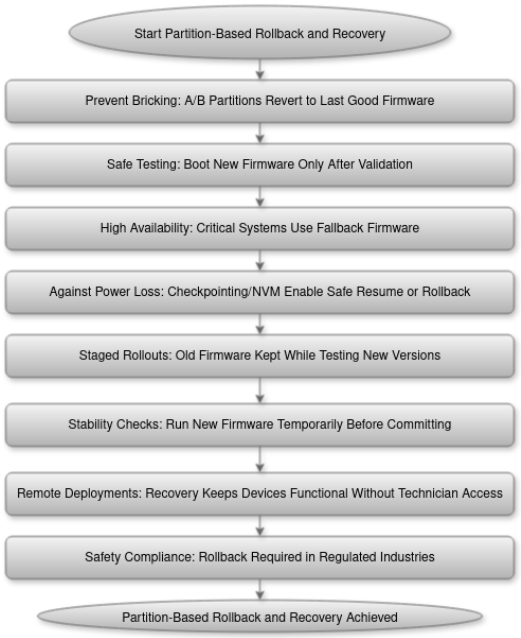
Technique T4.1:
Partition-Based Rollback and Recovery.

Description:
Use A/B partitioning, checkpointing, Non-Volatile Memory (NVM), and bootloader switching to enable firmware rollback and system recovery after failed updates.

General Use Cases:
- When preventing device bricking is essential, the technique reverts to the last known-good firmware through A/B partition fallback if an update fails.
- When testing new firmware safely, the technique boots the updated software only after successful validation and rolls back if verification fails.
- When high availability is required, the technique maintains system continuity by activating fallback firmware in the event of an update failure.
- When power loss occurs during an update, the technique ensures safe recovery through checkpointing and non-volatile memory that supports resume or rollback.
- When performing staged rollouts, the technique retains the old firmware across device populations, allowing new versions to be tested without risking a full deployment.
- When validating firmware stability, the technique temporarily installs new firmware before permanently committing to it.
- When devices operate in remote environments, the technique maintains functionality without technician access by using partition-based recovery.
- When meeting safety compliance requirements, the technique fulfills regulations by employing guaranteed rollback mechanisms.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Rollback prevents compromised firmware use.
- **Scalability**: (+1), Device populations supported by partitioned updates.
- **Performance**: (+1), Recovery time shortened by fallback.
- **Availability**: (+2), Fallback ensures uptime.
- **Interoperability**: (0), Rollback process device-specific.
- **Reliability**: (+2), A/B partitions prevent bricking.
- **Privacy**: (0), Rollback doesn't affect data use.
- **Energy Management**: (–1), Dual partitions increase storage/power use.
- **Flexibility**: (+1), Adaptable to system lifecycles.
- **Evolvability**: (+1), Staged rollouts extend lifecycle safety.

Technique Workflow Diagram:



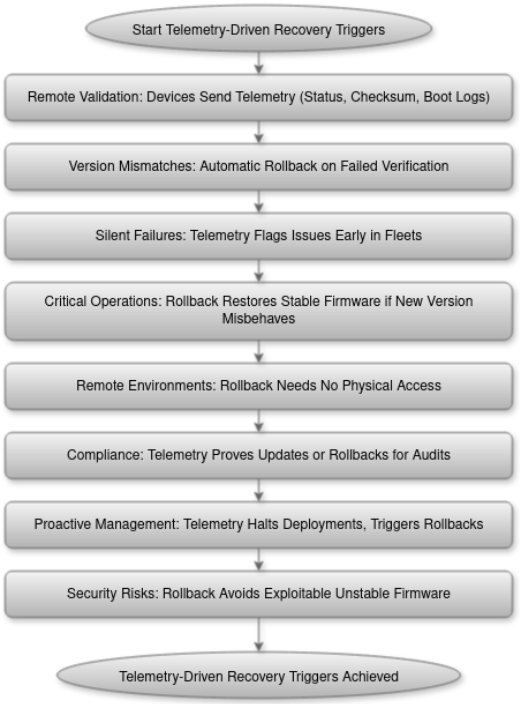
Technique T4.2:
Telemetry-Driven Recovery Triggers.

Description:
Monitor update outcomes via telemetry and version checks, initiating rollback if errors or inconsistencies are detected.

General Use Cases:
- When remote validation is required, the technique confirms update success by sending telemetry such as status, checksums, and boot logs.
- When version verification fails, the technique automatically triggers rollback to prevent mismatched firmware from running.
- When silent failures must be avoided, the technique detects issues early by monitoring telemetry across device populations.
- When devices perform critical operations, the technique restores stable firmware through rollback if the new version behaves incorrectly.
- When operating in remote environments, the technique enables recovery without physical access by using telemetry-triggered rollback.
- When compliance is necessary, the technique provides audit evidence by logging telemetry for successful updates and rollbacks.
- When proactive device population management is needed, the technique allows operators to halt deployments and initiate rollbacks based on telemetry signals.
- When security risks arise from unstable firmware, the technique prevents exploitation by reverting to a secure version through rollback mechanisms.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Logs and checksums ensure trusted rollback.
- **Scalability**: (+1), Telemetry manages device populations remotely.
- **Performance**: (0), Rollback decisions based on telemetry data.
- **Availability**: (+2), Rollback auto-triggers on failure.
- **Interoperability**: (0), Telemetry integration varies.
- **Reliability**: (+2), Telemetry confirms update success or failure.
- **Privacy**: (0), Neutral.
- **Energy Management**: (–1), Telemetry consumes resources.
- **Flexibility**: (+1), Adjustable rollback criteria.
- **Evolvability**: (+1), Adaptive rollback logic updates over time.

Technique Workflow Diagram:



Technique T4.3:
Role-Based Recovery Initiation.

Description:
Allow authorized roles, such as firmware uploader or manufacturer, to initiate recovery using dashboards or management tools.

General Use Cases:
- When trusted recovery is required, the technique prevents unauthorized actions by allowing only OEMs or administrators to trigger rollbacks.
- When remote restoration is needed, the technique enables authorized personnel to initiate recovery through dashboards and management platforms.
- When separation of duties is necessary, the technique prevents accidental or malicious recovery by distributing rollback authority across different roles.
- When failures affect large device populations, the technique supports coordinated recovery by allowing centralized initiation of group rollbacks.
- When accountability is essential, the technique provides auditability by logging rollback actions with role and identity details.
- When granular control is required, the technique limits recovery authorization to specific device groups or tenants based on policy.
- When complying with safety or regulatory requirements, the technique ensures that only high-privilege roles can perform critical recovery operations.
- When restoring remote deployments, the technique secures rollback procedures by enforcing role-based access without requiring physical presence.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Restricts rollback to authorized roles.
- **Scalability**: (+1), Centralized rollback for large device populations.
- **Performance**: (0), Role-based checks have minimal impact.
- **Availability**: (+1), Admins restore functionality quickly.
- **Interoperability**: (0), IAM integration required.
- **Reliability**: (+1), Prevents malicious or accidental recovery.
- **Privacy**: (+1), Logs track accountable actions.
- **Energy Management**: (0), No energy relation.
- **Flexibility**: (+1), Policies enforce role control.
- **Evolvability**: (+1), Roles adapt to organizational changes.

Technique Workflow Diagram:

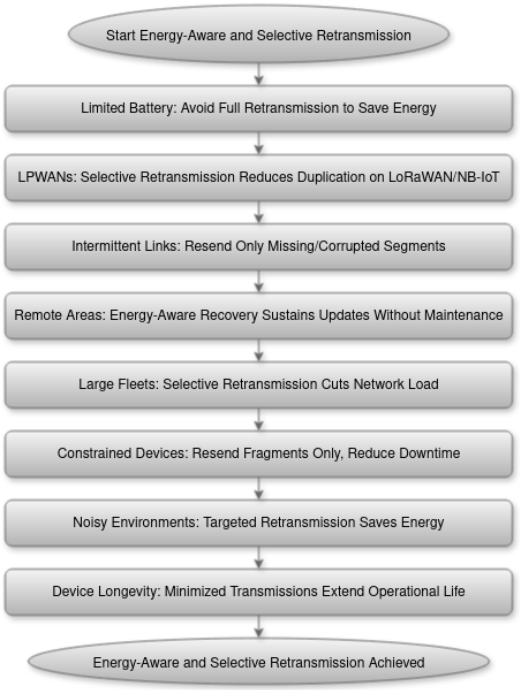
Technique T4.4:
Energy-Aware and Selective Retransmission.

Description:
Implement recovery strategies that minimize energy consumption by using selective retransmission for failed update segments.

General Use Cases:
- When battery capacity is limited, the technique saves energy and radio usage by avoiding full retransmission.
- When operating over LPWANs, the technique minimizes duplication on LoRaWAN and NB-IoT by selectively retransmitting only necessary data.
- When connectivity is intermittent, the technique improves efficiency by resending only missing or corrupted segments.
- When devices operate in remote areas, the technique enables continuous update processes without requiring frequent maintenance through energy-aware recovery.
- When updating large device populations, the technique reduces network load and energy consumption by using selective retransmission.
- When devices have constrained resources, the technique reduces downtime by retransmitting only required fragments.
- When networks experience interference or noise, the technique conserves energy by targeting retransmission to lost packets in lossy environments.
- When maximizing device longevity is important, the technique extends operational life by minimizing unnecessary transmissions.

Quality Attributes (QAs) Contribution:
- **Security**: (0), No direct protection.
- **Scalability**: (+1), Efficient retransmission supports device populations.
- **Performance**: (+1), Efficient use of bandwidth.
- **Availability**: (+1), Targeted retries reduce downtime.
- **Interoperability**: (0), Depends on network protocols.
- **Reliability**: (+1), Selective retransmission ensures data integrity.
- **Privacy**: (0), Not related.
- **Energy Management**: (+2), Retransmits only missing fragments.
- **Flexibility**: (+1), Supports constrained or remote environments.
- **Evolvability**: (+1), Adaptable to network and device conditions.

Technique Workflow Diagram:



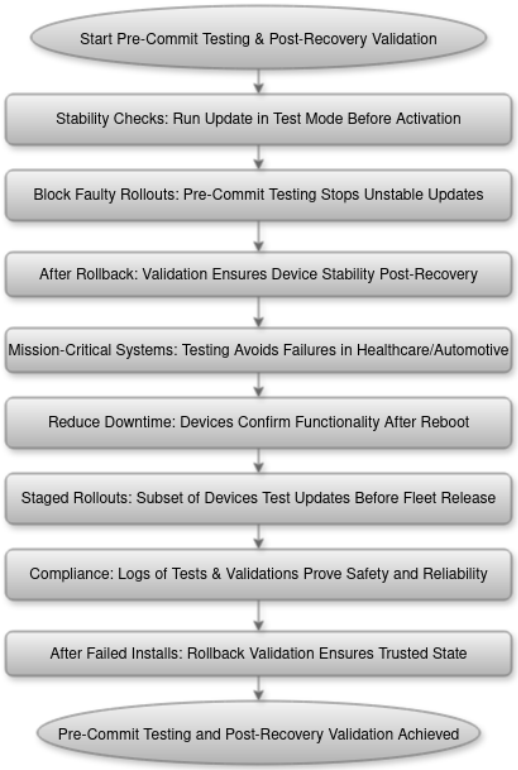
Technique T4.5:
Pre-Commit Testing and Post-Recovery Validation.

Description:
Conduct update code testing before committing changes and verify system stability after recovery to ensure proper functionality.

General Use Cases:
- When stability must be verified, the technique runs updates in a test mode before activation.
- When preventing faulty rollouts is critical, the technique blocks unstable updates through pre-commit testing.
- When a rollback occurs, the technique ensures device stability by validating the system after recovery.
- When operating in mission-critical domains such as healthcare or automotive, the technique avoids catastrophic failures through rigorous testing.
- When reducing downtime is important, the technique confirms device functionality immediately after reboot.
- When compliance is required, the technique proves safety and reliability by logging test and validation results.
- When installation fails, the technique ensures a trusted system state through rollback validation.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Prevents unsafe firmware activation.
- **Scalability**: (+1), Staged rollout with tests scales to device populations.
- **Performance**: (0), Neutral.
- **Availability**: (+2), Stable after rollback validation.
- **Interoperability**: (0), Validation process varies.
- **Reliability**: (+2), Test mode prevents unstable rollout.
- **Privacy**: (0), Not impacted.
- **Energy Management**: (–1), Tests consume resources.
- **Flexibility**: (+1), Configurable test and validation process.
- **Evolvability**: (+1), Supports adaptive staged strategies.

Technique Workflow Diagram:



Techniques Catalog: Mechanism for Updates Scheduling (M5)

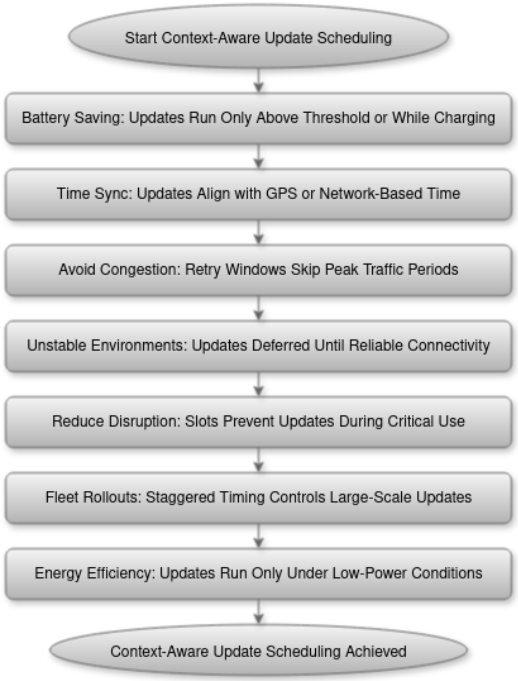
Technique T5.1:
Context-Aware Update Scheduling.

Description:
Schedule updates based on battery thresholds, GPS or network time synchronization, retry windows, and deferred execution slots.

General Use Cases:
- When battery preservation is required, the technique runs updates only above a power threshold or while charging.
- When accurate timing is needed, the technique aligns update execution with GPS or network-based time sources.
- When avoiding network congestion is necessary, the technique schedules retries outside peak traffic periods.
- When connectivity is unstable, the technique defers updates until a reliable connection is available.
- When minimizing disruption is important, the technique prevents updates during critical device use periods.
- When performing device population rollouts, the technique controls large-scale deployments through staggered timing.
- When energy efficiency is needed, the technique executes updates only under low-power operating conditions.

Quality Attributes (QAs) Contribution:
- **Security**: (0), Not directly related.
- **Scalability**: (+1), Staggered scheduling across device populations.
- **Performance**: (+1), Avoids congestion periods.
- **Availability**: (+2), Avoids downtime during critical tasks.
- **Interoperability**: (0), Not protocol-related.
- **Reliability**: (+2), Updates only when conditions are safe.
- **Privacy**: (0), No user data involved.
- **Energy Management**: (+2), Respects battery thresholds.
- **Flexibility**: (+1), Adjustable by context (time, power, connectivity).
- **Evolvability**: (+1), Adaptive to new constraints.

Technique Workflow Diagram:



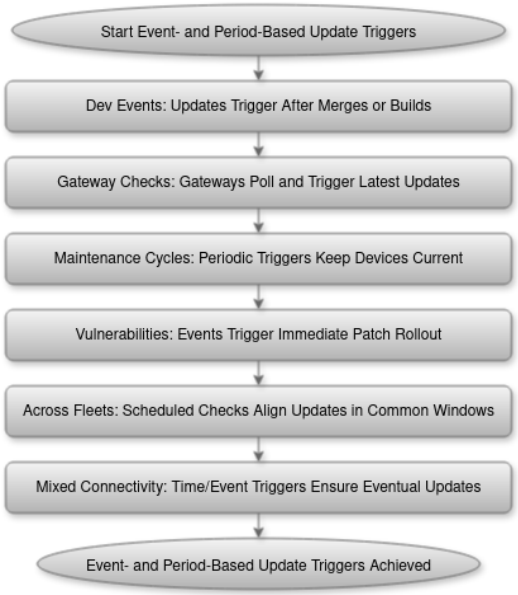
Technique T5.2:
Event- and Period-Based Update Triggers.

Description:
Initiate updates via specific events, such as repository merges, or through periodic checks of the gateway version.

General Use Cases:
- When development events occur, the technique triggers updates automatically after merges or builds.
- When gateways perform polling checks, the technique initiates the latest updates through gateway-triggered activation.
- When maintenance cycles are scheduled, the technique keeps devices current through periodic update triggers.
- When vulnerabilities are detected, the technique triggers the immediate deployment of patches to mitigate security risks.
- When connectivity conditions vary, the technique ensures that updates are eventually applied by combining time-based and event-driven triggers.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Faster patches on vulnerabilities.
- **Scalability**: (+1), Gateways handle triggers for device populations.
- **Performance**: (0), Neutral impact.
- **Availability**: (+1), Avoids drift with event triggers.
- **Interoperability**: (0), Neutral.
- **Reliability**: (+1), Consistent updates via periodic checks.
- **Privacy**: (0), No data protection function.
- **Energy Management**: (0), Not optimized for power.
- **Flexibility**: (+1), Supports periodic or event-driven updates.
- **Evolvability**: (+1), New events or policies easily added.

Technique Workflow Diagram:



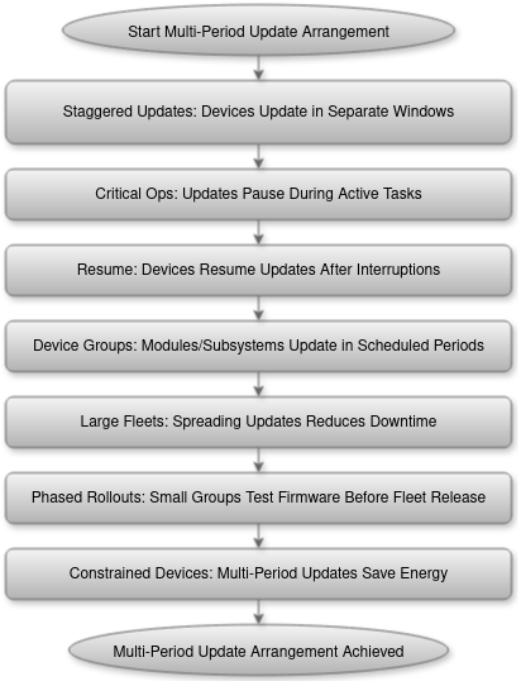
Technique T5.3:
Multi-Period Update Arrangement.

Description:
Organize updates into multiple periods with the ability to pause or resume specific update modules.

General Use Cases:
- When staggered updates are needed, the technique eases network load by updating devices in separate time windows.
- When devices perform critical operations, the technique prevents disruption by pausing updates until active tasks are completed.
- When interruptions occur, the technique ensures continuity by resuming updates from the last successful point.
- When managing device groups, the technique schedules updates for modules or subsystems during defined periods.
- When updating large device populations, the technique reduces downtime by spreading updates over a period of time.
- When devices are power-constrained, the technique saves energy by distributing updates across multiple low-power periods.

Quality Attributes (QAs) Contribution:
- **Security**: (0), No direct protection.
- **Scalability**: (+1), Spreads updates over time.
- **Performance**: (+1), Distributes load across time windows.
- **Availability**: (+2), Updates paused and resumed safely.
- **Interoperability**: (0), Neutral.
- **Reliability**: (+1), Reduces errors in staged deployments.
- **Privacy**: (0), Neutral.
- **Energy Management**: (+1), Avoids continuous usage spikes.
- **Flexibility**: (+2), Highly configurable (pause/resume per module).
- **Evolvability**: (+1), Supports phased rollouts.

Technique Workflow Diagram:



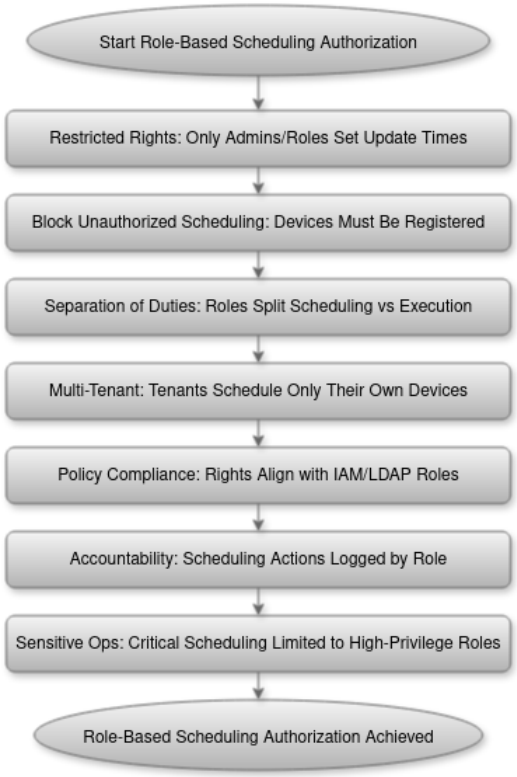
Technique T5.4:
Role-Based Scheduling Authorization.

Description:
Enforce scheduling permissions through dashboard role assignments and require device registration/authentication before scheduling.

General Use Cases:
- When update timing must be restricted, the technique ensures that only administrators or authorized roles can set schedules through the dashboard.
- When preventing unauthorized scheduling is required, the technique allows scheduling only for registered and authenticated devices.
- When separation of duties is necessary, the technique divides responsibilities so that one role schedules updates and another executes them.
- When operating in multi-tenant environments, the technique ensures that each tenant can schedule updates only for their own devices.
- When enforcing policy compliance, the technique aligns scheduling permissions with enterprise IAM or LDAP role definitions.
- When accountability is required, the technique records scheduling actions by role to support audit trails.
- When protecting sensitive operations, the technique limits critical scheduling capabilities to high-privilege roles.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Prevents unauthorized scheduling.
- **Scalability**: (+1), IAM and roles manage multi-tenant setups.
- **Performance**: (0), Neutral.
- **Availability**: (+1), Ensures critical systems remain online.
- **Interoperability**: (0), Depends on IAM integrations.
- **Reliability**: (+1), Avoids accidental or malicious schedules.
- **Privacy**: (+1), Roles limit exposure of sensitive information.
- **Energy Management**: (0), Neutral.
- **Flexibility**: (+1), Schedules per role or customer.
- **Evolvability**: (+1), Policies evolve with organizations.

Technique Workflow Diagram:



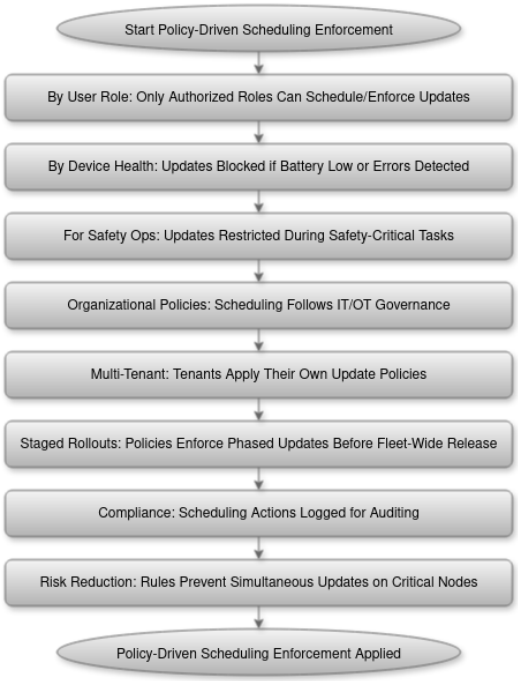
Technique T5.5:
Policy-Driven Scheduling Enforcement.

Description:
Enforce update scheduling and execution policies based on user role, device status, or operational safety constraints.

General Use Cases:
- When user-based control is required, the technique ensures only authorized roles can schedule or enforce updates.
- When device health is compromised, the technique blocks updates if the device has a low battery, errors, or instability.
- When devices perform safety-critical operations, the technique restricts updates to prevent disruption during sensitive tasks.
- When organizational governance applies, the technique enforces scheduling according to IT/OT policy requirements.
- When operating in multi-tenant environments, the technique ensures each tenant applies its own update policies independently.
- When auditability is required, the technique logs scheduling actions to support compliance and review.
- When reducing operational risk is necessary, the technique prevents simultaneous updates across critical nodes.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Strict role and device health rules.
- **Scalability**: (+1), Policies apply across tenants and device populations.
- **Performance**: (+1), Reduces risk of simultaneous updates.
- **Availability**: (+1), Prevents updates during critical tasks.
- **Interoperability**: (0), Neutral.
- **Reliability**: (+2), Blocks updates on unstable devices.
- **Privacy**: (0), No user data impact.
- **Energy Management**: (0), Neutral.
- **Flexibility**: (+1), Highly configurable policy enforcement.
- **Evolvability**: (+2), Policies evolve with governance.

Technique Workflow Diagram:



Techniques Catalog: Mechanism for Elaboration and Packaging (M6)

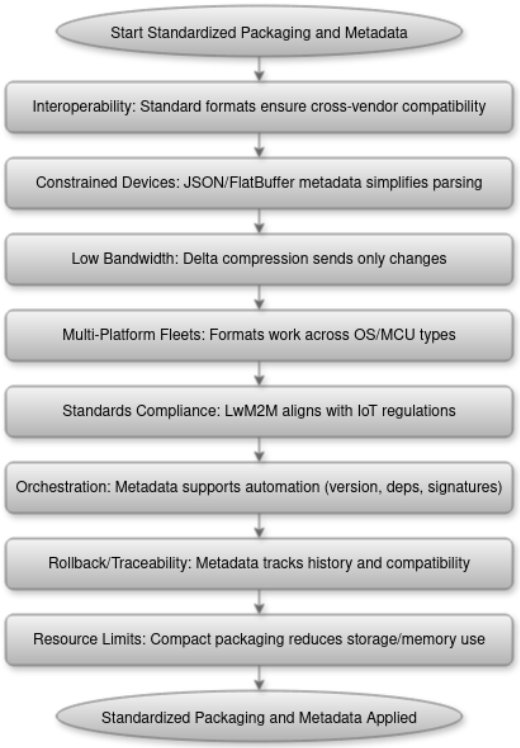
Technique T6.1:
Standardized Packaging and Metadata.

Description:
Use standard update formats (SWUpdate, LwM2M) with structured metadata (JSON, FlatBuffers) and delta compression for efficient update creation.

General Use Cases:
- When interoperability is required, the technique ensures cross-vendor compatibility by using standard formats such as .swu and LwM2M.
- When devices are constrained, the technique simplifies parsing by using lightweight metadata formats such as JSON or FlatBuffer.
- When bandwidth is limited, the technique saves cost and time by sending only changes through delta compression.
- When managing multi-platform device populations, the technique maintains compatibility across different OS and MCU types by using standardized formats.
- When regulatory compliance is necessary, the technique aligns with IoT standards by implementing LwM2M.
- When orchestrating OTA processes, the technique supports automation by including metadata such as version, dependencies, and signatures.
- When rollback and traceability are required, the technique preserves history and compatibility information through structured metadata.
- When resources are limited, the technique reduces storage and memory usage through compact packaging.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Metadata and signatures improve authenticity.
- **Scalability**: (+1), Common formats ease device population orchestration.
- **Performance**: (+1), Delta compression improves efficiency.
- **Availability**: (0), Not directly improved.
- **Interoperability**: (+2), Standard formats enable cross-vendor use.
- **Reliability**: (+1), Metadata tracks compatibility and history.
- **Privacy**: (0), No personal data effect.
- **Energy Management**: (0), Neutral.
- **Flexibility**: (+1), Metadata supports automation and adaptability.
- **Evolvability**: (+1), Metadata supports long-term updates.

Technique Workflow Diagram:



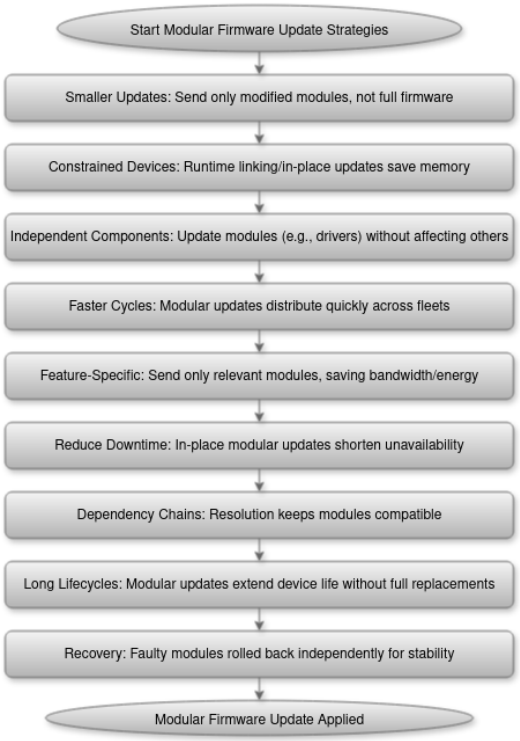
Technique T6.2:
Modular Firmware Update Strategies.

Description:
Implement runtime linking, in-place updates, and dependency resolution to enable modular and flexible firmware updates.

General Use Cases:
- When smaller updates are needed, the technique minimizes overhead by sending only modified modules instead of full firmware images.
- When devices have constrained memory, the technique saves resources by using runtime linking and in-place updates.
- When components are independent, the technique updates individual modules (e.g., drivers) without affecting the rest of the system.
- When fast update cycles are required, the technique accelerates deployment by distributing modular updates quickly across device populations.
- When feature-specific changes are required, the technique conserves bandwidth and energy by sending only the relevant modules.
- When software dependency chains exist, the technique maintains stability by resolving module compatibility before installation.
- When devices require long operational lifecycles, the technique extends longevity by updating modules instead of replacing the entire system.
- When recovering from faults, the technique restores stability by rolling back only the affected modules independently.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Module isolation improves trust.
- **Scalability**: (+1), Efficient device population-wide updates.
- **Performance**: (+1), Smaller modules reduce cost.
- **Availability**: (+1), Shorter downtime via in-place updates.
- **Interoperability**: (0), Varies across devices.
- **Reliability**: (+1), Faulty modules roll back independently.
- **Privacy**: (0), Not impacted.
- **Energy Management**: (+1), Smaller payloads save power.
- **Flexibility**: (+2), Independent modules updated separately.
- **Evolvability**: (+2), Modular updates extend lifecycle.

Technique Workflow Diagram:



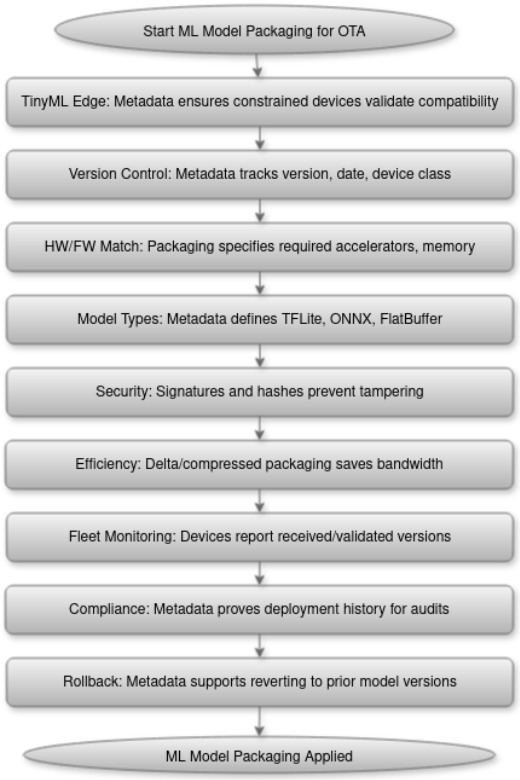
Technique T6.3:
ML Model Packaging for OTA.

Description:
Package OTA-delivered machine learning models with embedded metadata to ensure compatibility and traceability.

General Use Cases:
- When running TinyML on edge devices, the technique ensures compatibility by using metadata that validates model requirements on constrained hardware.
- When version control is needed, the technique tracks firmware/model evolution through embedded metadata, including version, date, and device class.
- When matching hardware and firmware, the technique specifies required capabilities (e.g., accelerators, memory) through packaging metadata to ensure proper execution.
- When supporting diverse model formats, the technique enables correct parsing by defining the model type (e.g., TFLite, ONNX, FlatBuffer) in metadata.
- When securing model delivery, the technique prevents tampering by enforcing signatures and cryptographic hashes.
- When optimizing OTA efficiency, the technique reduces bandwidth usage by applying delta and compressed model packaging during retraining cycles.
- When monitoring device populations, the technique ensures visibility and traceability by having devices report received and validated model versions.
- When compliance requirements apply, the technique supports audits by providing metadata that records deployment history.
- When rollback is necessary, the technique enables recovery by using metadata to revert devices to prior model versions.

Quality Attributes (QAs) Contribution:
- **Security**: (+1), Signatures and hashes protect model integrity.
- **Scalability**: (+1), Device populations can validate and deploy models.
- **Performance**: (+1), Compressed packaging reduces cost.
- **Availability**: (+1), Targeted updates for ML-driven functions.
- **Interoperability**: (+1), Standardized formats (ONNX, TFLite).
- **Reliability**: (+1), Metadata ensures compatibility.
- **Privacy**: (0), Not directly improved.
- **Energy Management**: (0), Neutral.
- **Flexibility**: (+1), Supports rollback and version control.
- **Evolvability**: (+2), Supports retraining and new ML models.

Technique Workflow Diagram:



Technique T6.4:
Storage Optimization via Compression and Caching.

Description:
Use compression and flash-based caching to minimize storage usage during updates.

General Use Cases:
- When flash storage is limited, the technique fits firmware within available space by compressing it (e.g., using gzip).
- When updates are large, the technique reduces bandwidth consumption and transfer time through compressed packages.
- When devices operate on low power, the technique saves energy by minimizing flash writes.
- When using cache-assisted installs, the technique stages updates in chunks using flash caching.
- When recovery is needed, the technique resumes updates without a full re-download by using cached data.
- When applying incremental updates, the technique efficiently stores partial deltas through compressed caching.
- When updating large device populations, the technique lowers server and network load by using compression and caching.
- When device longevity is a priority, the technique extends hardware lifespan by reducing write cycles through caching.
- When networks are constrained, the technique enables OTA delivery over low-speed links through compressed transfers.

Quality Attributes (QAs) Contribution:
- **Security**: (0), Neutral.
- **Scalability**: (+1), Efficient use of storage across device populations.
- **Performance**: (+2), Compressed transfers speed up.
- **Availability**: (+1), Staged caching reduces downtime.
- **Interoperability**: (0), Depends on storage setup.
- **Reliability**: (+1), Cached updates recover without re-download.
- **Privacy**: (0), Neutral.
- **Energy Management**: (+2), Fewer flash writes save power.
- **Flexibility**: (0), Minimal adaptability change.
- **Evolvability**: (+1), Caching supports flexible updates.

Technique Workflow Diagram:

Technique T6.5:
Secure and Optimized Payload Formats.

Description:
Apply compression, delta updates, or custom payload formats with integrated versioning and encryption for secure, efficient delivery.

General Use Cases:
- When reducing payload size is required, the technique saves bandwidth and accelerates delivery through delta and compression mechanisms.
- When firmware confidentiality is critical, the technique prevents disclosure and reverse-engineering by encrypting payloads.
- When maintaining version control, the technique enforces correctness and prevents downgrades through integrated versioning.
- When links are constrained, the technique optimizes LPWAN and satellite transmissions by using compact payload formats.
- When performing modular updates, the technique reduces data transfer by delivering only the necessary modules through segmented payloads.
- When ensuring device population consistency, the technique keeps devices aligned by distributing version-tagged payloads.
- When securely staging updates, the technique protects cached data by using encrypted and compressed formats.
- When compliance requirements apply, the technique meets standards by distributing structured payloads with metadata.
- When improving error resilience, the technique recovers corrupted data by validating each segment with checksums and modular verification.

Quality Attributes (QAs) Contribution:
- **Security**: (+2), Encryption and integrity checks secure updates.
- **Scalability**: (+1), Compact payloads scale to device populations.
- **Performance**: (-1), Compression and deltas overhead.
- **Availability**: (+1), Segmented payloads ensure continuity.
- **Interoperability**: (+1), Structured payloads improve compatibility.
- **Reliability**: (+1), Modular validation ensures safe installs.
- **Privacy**: (0), Not directly protected.
- **Energy Management**: (+1), Smaller payloads conserve power.
- **Flexibility**: (+1), Modular delivery per feature.
- **Evolvability**: (+1), Versioning avoids downgrade issues.

Technique Workflow Diagram: