FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Democratization of Technical Development in Business Environments

João Tomás Marques Félix

Mestrado em Engenharia Informática e Computação

Supervisor: Professor Doutor Nuno Honório Rodrigues Flores

July 29, 2025

Democratization of Technical Development in Business Environments

João Tomás Marques Félix

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Professora Doutora Mariana Curado Malta
Referee: Professora Doutora Ana Isabel Rojão Lourenço Azevedo
Referee: Professor Doutor Nuno Honório Rodrigues Flores

July 29, 2025

Resumo

A crescente complexidade dos ecossistemas tecnológicos em grandes organizações tornou a integração de sistemas uma peça fundamental para a agilidade, escalabilidade e inovação do negócio. Tradicionalmente, esta responsabilidade recaía sobre equipas centralizadas (*Center of Excellence, CoE*), criando pontos de fricção e limitando a autonomia das equipas de negócio. Este trabalho propõe e avalia a transição para um modelo descentralizado de integração — o *Center for Enablement (C4E)* — suportado pela criação de dois artefactos: uma framework automatizada de *Event-Driven Architecture (EDA)* e uma plataforma moderna de *Managed File Transfer (MFT)*, ambos implementados na MC Digital, empresa tecnológica da Sonae MC.

A framework EDA desenvolvida permite que as equipas técnicas e não-técnicas definam e implementem fluxos de eventos de forma autónoma, através de especificações YAML, validação automática, provisionamento programático de recursos Kafka e documentação integrada. A solução MFT substitui a abordagem SOA legada por uma pipeline declarativa, segura e auditável, que suporta transferências de ficheiros entre sistemas internos, parceiros externos e plataformas *cloud*. Os resultados quantitativos mostram reduções drásticas nos tempos de transferência, enquanto a análise qualitativa, baseada em personas e feedback real, comprova o aumento da autonomia, satisfação e colaboração entre as equipas.

A investigação conclui que a combinação de self-service, automação e validação rigorosa permite democratizar a integração de sistemas sem comprometer a governação ou a segurança, promovendo uma cultura de melhoria contínua e inovação sustentável. O trabalho apresenta um modelo replicável para a transformação digital de organizações de grande escala, alinhando-se com as melhores práticas de *governance* de tecnologias de informação.

Palavras-chave: integração de sistemas, *event-driven architecture*, *managed file transfer*, automação, self-service, governabilidade, C4E, transformação digital.

Abstract

The increasing complexity of technological ecosystems in large organizations has made system integration a critical driver of business agility, scalability, and innovation. Traditionally, this responsibility fell to centralized teams (Centers of Excellence, CoE), resulting in bottlenecks and limiting the autonomy of business units. This dissertation proposes and evaluates a transition to a decentralized integration model—Center for Enablement (C4E)—through the design and implementation of two artefacts: an automated Event-Driven Architecture (EDA) framework and a modern Managed File Transfer (MFT) platform, both deployed at MC Digital, the technology arm of Sonae MC.

The developed EDA framework enables both technical and non-technical teams to autonomously define and deploy event flows using YAML specifications, automated validation, programmatic Kafka resource provisioning, and integrated documentation. The MFT solution replaces the legacy SOA-based approach with a declarative, secure, and auditable pipeline, supporting file transfers between internal systems, external partners, and cloud platforms. Quantitative results demonstrate significant reductions in onboarding and transfer times, while qualitative analysis, grounded in user personas and real-world feedback, confirms increased autonomy, satisfaction, and collaboration across teams.

This research concludes that combining self-service, automation, and rigorous validation democratizes system integration without compromising governance or security, fostering a culture of continuous improvement and sustainable innovation. The work provides a replicable model for digital transformation in large-scale organizations, aligned with international best practices in information technology governance.

Keywords: system integration, event-driven architecture, managed file transfer, automation, self-service, governance, C4E, digital transformation.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

This dissertation addresses the growing need for digital transformation and efficiency in large organizations, focusing on the democratization of system integration through automation and self-service frameworks. Although not directly focused on environmental or social intervention, the work supports organizational and operational improvements that are indirectly aligned with several SDGs, especially those related to innovation, sustainable economic growth, industry resilience, and fostering inclusive digital transformation.

The main SDGs addressed are:

SDG 8: Decent Work and Economic Growth — Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all.

SDG 9: Industry, Innovation and Infrastructure — Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation.

SDG 12: Responsible Consumption and Production — Ensure sustainable consumption and production patterns (insofar as improved automation and governance reduce waste, rework, and resource inefficiency in IT operations).

Within these SDGs, the dissertation contributes in particular to the following targets:

Target 8.2 — Achieve higher levels of economic productivity through diversification, technological upgrading and innovation.

Target 9.5 — Enhance scientific research, upgrade the technological capabilities of industrial sectors, and encourage innovation.

Target 9.c — Significantly increase access to information and communications technology and strive to provide universal and affordable access to the internet.

Target 12.6 — Encourage companies, especially large and transnational companies, to adopt sustainable practices and to integrate sustainability information into their reporting cycle.

SGD	Target	Contribution	Performance Indicators and Metrics
8	8.2	Increases productivity through digital transformation and process automation.	Decrease in average onboarding time for new projects.
9	9.5, 9.c	Fosters innovation, digital skills, and access to IT for more teams.	Number of teams onboarded; increase in integrations managed by business users.
12	12.6	Promotes efficient and responsible use of IT resources through automation and reduced manual rework.	Reduction in manual errors; decrease in support interventions.

In summary, while the dissertation is focused on technological and organizational innovation within a business context, its contributions to digital transformation, efficiency, and inclusive access to technology are closely aligned with the United Nations SDGs for industry, innovation, and sustainable economic growth. This work demonstrates how engineering solutions can have a broader positive impact on society, supporting progress toward a more inclusive and sustainable future.

Acknowledgements

This dissertation marks the culmination of five years of significant learning and personal growth. For this reason, I would like to express my sincere gratitude to:

Professor Dr.Nuno Flores, for all his guidance, support, and availability throughout this journey.

Flávio Saraiva, Pedro Gomes, Tiago Augusto and José Leal, for their support and for helping to clarify the entire process throughout this project, as well as for providing me with the essential foundations.

My team at Sonae, for warmly welcoming me and providing the ideal environment to carry out this work.

My parents, for the sacrifices they have always made for me, and for their unwavering support throughout my academic journey and every life choice I have made.

My grandparents, for their affection and for the values and teachings that have continuously shaped me.

My godmother, my godfather, Sérgio, and Joana, for being a constant example and source of inspiration in my life.

My cousins, for being the siblings I never had and for their constant support along this journey.

Teresa, for her patience, kindness, wise words, and the compassion she brings into every day. You continuously inspire me to become a better person.

My friends, for all the memories we built together, the laughter we shared, the help offered whenever needed, and for everything that is still to come.

To all of you, my heartfelt thank you. It is only because of you that all of this truly makes sense.

"Believe you can and you're halfway there"

Theodore Roosevelt

Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
C4E	Center for Enablement
CI/CD	Continuous Integration/Continuous Deployment
CoE	Center of Excellence
DSR	Design Science Research
DevOps	Development Operations
EDA	Event-Driven Architecture
IaC	Infrastructure as Code
IT	Information Technology
LoB	Line of Business
MFT	Managed File Transfer
SOA	Service-Oriented Architecture
YAML	YAML Ain't Markup Language

Chapter 1

Introduction

This document constitutes a Master's dissertation in Informatics and Computing Engineering at the Faculty of Engineering of the University of Porto (FEUP). It presents the results of an academic and professional research project conducted within the MC Digital division of Sonae MC.

The scope of this dissertation encompasses the identification, design, implementation, and evaluation of technical solutions to contemporary challenges in enterprise system integration. The title, "Democratization of Technical Development in Business Environments," reflects the core goal of this work: to enable a much wider set of teams and individuals—including those with limited prior experience in integration technologies—to independently design and implement integration patterns. By democratizing technical development, the proposed solutions simplify complex integration processes, providing user-friendly, guided frameworks that allow even non-specialist users to participate fully in building, validating, and deploying integration flows. This empowerment of business teams reduces dependency on central IT experts, accelerates innovation, and ensures that technical development is accessible to all, truly embodying the notion of democratization within modern enterprise environments.

1.1 Enterprise Landscape

Sonae

Sonae is a leading Portuguese multinational group with a diversified portfolio operating across various economic sectors, including food and non-food retail, real estate, financial services, telecommunications, technology, and investment management. Founded in 1959 and headquartered in Maia, Sonae has evolved into one of Portugal's largest private sector employers, with operations extending across Europe, South America, and Africa. The group's strategy has long been characterized by a focus on innovation, digital transformation, and operational excellence, leveraging technology as a core enabler for its businesses. Sonae's commitment to sustainability, entrepreneurship, and continuous improvement has positioned it as a reference point in the Portuguese and international business landscape.

Sonae MC

Sonae MC (Modelo Continente) is the food retail arm of the Sonae group and the market leader in the Portuguese retail sector. With a network that includes hypermarkets, supermarkets, convenience stores, and specialty shops—such as Continente, Continente Bom Dia, and Wells—Sonae MC manages a vast ecosystem of physical and digital channels. The company is recognized for its pioneering role in the adoption of innovative business models and digital technologies to enhance customer experience, operational efficiency, and supply chain integration. Sonae MC's large-scale operations, complex logistics, and dynamic business requirements make integration technology a fundamental pillar for ensuring agility, resilience, and continued leadership in a highly competitive market.

MC Digital

MC Digital is the technology and digital solutions company within Sonae MC, acting as both a strategic enabler and an operational backbone for all business units under Sonae MC. Established to accelerate the group's digital transformation, MC Digital is responsible for architecting, delivering, and continuously evolving the mission-critical IT platforms that power Sonae MC's retail, supply chain, and customer engagement activities.

As the digital and technological center of excellence for Sonae MC, MC Digital provides a comprehensive portfolio of services, ranging from software engineering, enterprise architecture, and integration platforms to data analytics, cloud infrastructure, cybersecurity, and DevOps. The company's teams collaborate closely with business stakeholders, translating strategic objectives into digital solutions that drive efficiency, innovation, and value creation across the organization.

A key part of MC Digital's mandate is to ensure the seamless integration of hundreds of applications and services, both internally and with external partners and suppliers. This is achieved through the governance, evolution, and support of robust integration platforms—spanning both event-driven and file-based paradigms—that enable real-time data exchange, process automation, and end-to-end visibility across the business.

The work described in this dissertation was carried out within the Integration Platforms team at MC Digital. This team is responsible for managing the full lifecycle of integration technologies: from defining architecture and standards to onboarding new projects, developing automation pipelines, and providing 24/7 operational support. The team's remit covers the deployment and governance of platforms such as Apache Kafka for event-driven integration, GoAnywhere for managed file transfer, and the implementation of automation and self-service frameworks that enable business teams to independently manage integration flows with full compliance and security.

By situating the dissertation in MC Digital, the research addresses not only theoretical challenges but also delivers practical, replicable solutions tailored to the realities of a large, complex, and digitally mature retail organization. The work thus contributes directly to Sonae MC's strategic ambition to be a digital leader, both in Portugal and internationally.

1.2 Context and Motivation

Integrating heterogeneous technical systems remains a central challenge for growing organizations, given the diversity of technologies, protocols, and data formats in modern enterprise IT landscapes. Traditionally, integration has been managed by centralized, highly specialized teams under the Center of Excellence (CoE) model (Fishman, 2022), ensuring standardization but often creating bottlenecks that slow innovation and responsiveness.

To address these limitations, many organizations are moving toward a decentralized Center for Enablement (C4E) model (Fishman, 2022), distributing responsibility to Lines of Business (LoBs) (Ovington, 2023) and equipping them with frameworks and governance to independently manage integration tasks. While this shift increases agility and scalability, it also introduces challenges in maintaining quality and governance, particularly when technical proficiency varies across business teams.

This dissertation focuses on democratizing two key enterprise integration patterns—event-driven architectures (EDA) (Seetharamugn, 2023) and file-based transfers—by developing automated, self-service frameworks. Event-driven integration, previously slowed by sequential workflows and central team dependencies, is streamlined by enabling LoBs to define event flows through structured, validated specifications. Similarly, Managed File Transfer (MFT)[1] replaces legacy SOA[2]-based file exchange with self-service, automated solutions that allow business teams to configure and manage file transfers independently, with built-in validation and security.

Ultimately, this work aims to demonstrate how decentralizing integration through automation and self-service can enhance efficiency, scalability, and responsiveness in large enterprises, while maintaining robust governance. The research addresses whether this transition from CoE to C4E—implemented through automated pipelines and self-service frameworks—can deliver tangible improvements in agility and delivery speed without sacrificing quality and compliance.

1.3 Goal and Methodology

The main research question of this dissertation, **"How does the transition from a centralized CoE to a decentralized C4E model, through the implementation of automated pipelines and self-service frameworks for event-driven and file-based integration, impact efficiency, scalability, and governance in large-scale enterprise environments?"** is addressed through a set of focused objectives, each corresponding to a dimension of organizational and technical change.

To provide actionable answers, the following concrete objectives have been defined:

1. **Design and implement an automated self-service framework for EDA**, enabling Lines of Business (LoB) teams to define and manage their own event flows with minimal dependency on central integration teams.

[1] https://www.goanywhere.com/managed-file-transfer (last accessed on Jun, 2025)
[2] https://aws.amazon.com/pt/what-is/service-oriented-architecture/ (last accessed on Jun, 2025)

2. **Develop and operationalize a modern MFT solution**, supporting decentralized onboarding and maintenance of file-based integration processes, with robust validation and governance.

3. **Establish and enforce automated validation, documentation, and access control mechanisms** across both EDA and MFT frameworks to ensure compliance, security, and maintainability, even as responsibility shifts to business teams.

4. **Evaluate the impact of the transition from CoE to C4E** on key performance indicators such as delivery lead time, scalability, operational effort, and governance effectiveness, using both quantitative (performance metrics, throughput, onboarding time) and qualitative (user feedback, process compliance) data.

To achieve these objectives and rigorously assess the impact of the proposed transition, this dissertation adopts the Design Science Research (DSR) methodology. DSR is a structured, iterative approach that emphasizes the design, development, and evaluation of innovative artefacts to solve real-world organizational problems. The choice of DSR ensures that both the frameworks and the results presented are not only practically relevant but also scientifically grounded. A detailed discussion of DSR and its application to this research is presented in a subsequent chapter.

1.4 Dissertation Structure

This dissertation is organized into five chapters, each building on the previous to provide a coherent and comprehensive exploration of the research question.

- **Chapter 1 – Introduction:** Introduces the research context, defines the motivation, establishes the research question, and presents the goals and methodology that guide the work.
- **Chapter 2 – State of the Art:** Provides a comprehensive literature review of the relevant concepts, including MFT, EDA, organizational integration patterns (CoE and C4E), and the governance and automation paradigms that support the proposed model.
- **Chapter 3 – Problem and Solution Approach:** Defines the functional and non-functional requirements based on the research question and proposes a conceptual integration model that aligns EDA and MFT within a C4E organizational structure.
- **Chapter 4 – Development and Implementation:** Details the technological stack and implementation strategy used to materialize the proposed model, including infrastructure setup, CI/CD pipeline automation, security governance, and monitoring. Also includes concrete use cases to validate the model in practice.
- **Chapter 5 – Results and Discussion:** Presents an evaluation of the solution's effectiveness in meeting the defined requirements. It also analyses stakeholder personas and the perceived added value of the model.

- **Chapter 6 – Conclusion:** Summarizes the key contributions of the dissertation, reflects on the limitations, and outlines opportunities for future research and development.

Chapter 2

State of The Art

The "State-of-the-Art" section will provide an in-depth analysis of this dissertation's theoretical and practical foundations. It will begin by contextualizing the evolution of system integration practices in modern organizations, highlighting the anticipated transition from centralized CoE models to a decentralized C4E frameworks. The chapter will then explore the expected critical role of automation and EDAs in facilitating this shift, particularly through platforms like Kafka[1] (Narkhede et al., 2011) and GoAnywhere.

Following this, the chapter will identify the anticipated challenges of implementing a C4E model, such as balancing governance with agility and addressing technical gaps within decentralized teams. Lastly, it will define the objectives of this dissertation, emphasizing the development of automated pipelines to streamline resource management, empower business teams, and maintain governance, scalability, and quality. This chapter will set the stage for understanding how these advancements are expected to contribute to the democratization of technical development in enterprise environments.

2.1 From CoE to C4E

Modern organizations face increasing complexity in their IT ecosystems due to the proliferation of diverse technologies, protocols, and data formats (Dhamdhere, 2020). Effective system integration is essential to enable heterogeneous applications, platforms, and data streams to interoperate, supporting business agility and innovation. Traditionally, this complexity led to the creation of specialized, centralized integration teams, typically organized under the CoE model. In the CoE paradigm, a single expert group is responsible for defining standards, building and maintaining integration logic, and ensuring compliance and security across the organization.

While CoE structures deliver high levels of consistency, governance, and technical quality, they also introduce significant challenges. As the volume, diversity, and velocity of integration requests increase, centralized teams may become a bottleneck, unable to respond quickly enough to evolving business needs. This is especially problematic in environments characterized by rapid

[1] https://kafka.apache.org/ (last accessed on Jun, 2025)

digital transformation, where responsiveness and innovation are paramount. Moreover, the sequential, project-based workflows typical of CoE models can result in long lead times, redundant coordination, and reduced flexibility for business units that need to react independently to market changes.

In response to these limitations, many enterprises are shifting towards the C4E model. Rather than concentrating integration responsibility within a single team, the C4E approach distributes knowledge, frameworks, and tools across Lines of Business (LoBs), empowering them to independently manage their own integration needs. The C4E model is underpinned by standardized automation, validation pipelines, and governance mechanisms that ensure compliance and quality even as operational responsibility becomes decentralized. This approach offers significant advantages in terms of agility, scalability, and alignment between IT and business objectives, as it enables business teams to accelerate delivery without waiting for central prioritization.

The transition from CoE to C4E has been further catalyzed by the adoption of modern integration patterns, notably EDA. The rise of platforms such as Apache Kafka has transformed how organizations approach integration, facilitating real-time, scalable, and decoupled communication between distributed systems (Narkhede et al., 2011). Automated pipelines for Kafka resource creation—such as those implemented in this project—illustrate the practical mechanisms by which the gap between centralized governance and decentralized enablement can be bridged. These pipelines enable business teams to provision events, configure access controls, and update event catalogs in a self-service manner, with built-in validation ensuring adherence to global standards (see Figure 2.1).

As a result, the shift from CoE to C4E does not imply the loss of control or governance, but rather a rebalancing of responsibilities: central teams continue to define policies and frameworks, while LoB teams are equipped with the autonomy and tools needed to innovate rapidly and securely. This model, when supported by robust automation and clear governance, is a key enabler for the democratization of IT and the acceleration of digital transformation in large, complex enterprises.

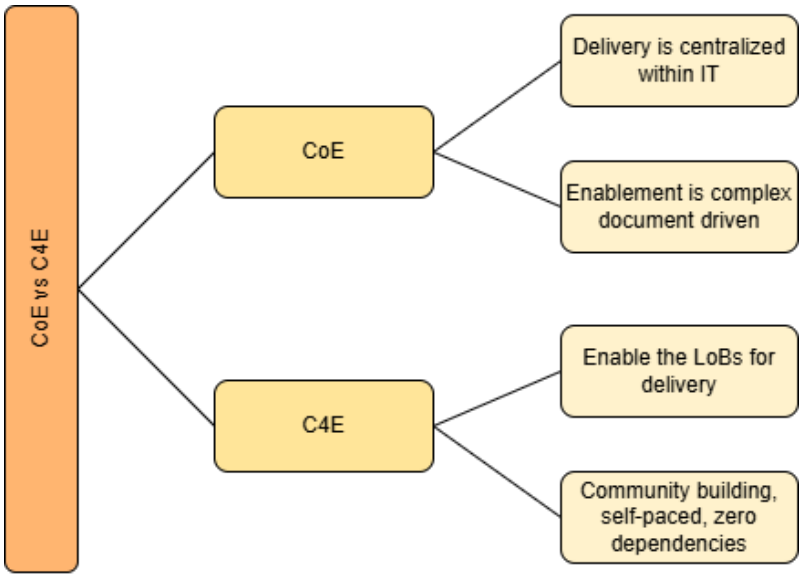


Figure 2.1: Comparison between CoE and C4E models.

2.2 Event-Driven Architectures (EDA)

EDA has emerged as a foundational paradigm in modern enterprise integration, enabling organizations to design highly scalable, loosely coupled, and reactive systems. At its core, EDA is based on the production, detection, and reaction to events—discrete state changes that signal meaningful occurrences within or between business systems (Narkhede et al., 2011). Unlike request-response models, EDA enables asynchronous communication, where producers emit events without direct knowledge of the consumers, fostering independence between system components.

This architectural approach is particularly well-suited to supporting digital transformation, business agility, and real-time analytics. By decoupling producers and consumers, EDA makes it easier to evolve systems incrementally, integrate new services without disrupting existing workflows, and rapidly adapt to changing requirements or business opportunities. Use cases range from microservices coordination, customer interaction tracking, fraud detection, IoT telemetry, and supply chain monitoring, to large-scale data ingestion for analytics platforms.

A central enabler of EDA in enterprise contexts is the adoption of scalable, high-throughput event streaming platforms such as Apache Kafka (Narkhede et al., 2011). Kafka provides durable, distributed, and fault-tolerant log storage, supporting millions of events per second and a variety of messaging patterns, including publish/subscribe, event sourcing, and stream processing. Its architecture allows for horizontal scalability, data retention, and exactly-once processing semantics—features critical for reliable enterprise-grade integration.

In the context of decentralized IT models such as C4E, EDA is a natural fit. LoB teams can independently publish and subscribe to events, develop microservices or analytics pipelines, and integrate new applications rapidly, provided organizational governance is enforced through automated validation, schema registry management, and event cataloging.

However, EDA also introduces new challenges: governance of schema evolution, versioning, topic proliferation, and secure multi-tenant access become critical as adoption scales. Ensuring data quality, maintaining event lineage, and monitoring event flow health are all necessary for operational maturity. Addressing these challenges requires a combination of automation, tooling, and a strong governance model aligned with C4E principles, as recognized by both academic research and industry best practices (Bhat & H, 2024; Bruns & Dunkel, 2014; Molina et al., 2018).

In summary, EDA is not only a technological solution but also an organizational enabler, allowing enterprises to democratize integration, drive innovation, and build for resilience in a data-driven world.

2.3 MFT and Legacy Integration

Despite the accelerating adoption of APIs and event-driven solutions, file-based integration remains deeply embedded in the digital backbone of many enterprises—particularly those with significant legacy infrastructure, batch data processes, or regulatory compliance requirements (Hyvönen et al., 2018; Krishnan et al., 2022). File transfer is often the default mechanism for exchanging large, structured datasets across disparate systems, business units, and external partners. Common examples include financial reporting, inventory updates, and supply chain data exchange.

Historically, these file flows were orchestrated using tightly coupled Service-Oriented Architecture (SOA) applications, custom scripts, or point-to-point integrations. While robust and secure, such models tend to be rigid, slow to adapt, and heavily reliant on specialized central IT teams for maintenance and change management. The lack of automation, limited error recovery, and insufficient auditability often result in operational bottlenecks, increased support effort, and exposure to compliance risks.

The evolution towards MFT platforms represents a significant modernization of this integration pattern. MFT solutions provide a secure, auditable, and standardized approach to file movement, supporting a broad range of protocols (FTP, SFTP), cloud storage backends (Azure Blob Storage[2], Amazon S3[3]), and enterprise authentication methods (Chukhajyan, 2016; Folds & McDermott, 2019). Modern MFT platforms automate the scheduling, routing, encryption, and monitoring of file transfers, while providing detailed audit trails, alerting, and self-service dashboards for business users.

MFT platforms also facilitate hybrid and multi-cloud integrations, bridging on-premises legacy systems with modern SaaS[4] and cloud storage[5], and supporting a smooth migration path for organizations modernizing their integration landscape. Nevertheless, the persistence of legacy protocols, diverse stakeholder requirements, and the criticality of file flows for core business processes mean that robust governance, automation, and monitoring remain essential. As such, MFT is not

[2] https://azure.microsoft.com/en-us/products/storage/blobs (last accessed on Jun, 2025)
[3] https://aws.amazon.com/s3/ (last accessed on Jun, 2025)
[4] https://azure.microsoft.com/pt-pt/resources/cloud-computing-dictionary/what-is-saas (last accessed on Jun, 2025)
[5] https://aws.amazon.com/what-is/cloud-storage/ (last accessed on Jun, 2025)

simply a "legacy holdover," but an active and strategic component of a comprehensive, democratized integration strategy for large-scale enterprises.

2.4 Self-Service and Automation in Integration

A cornerstone of the C4E paradigm is self-service, achieved through declarative configuration, automation pipelines, and robust validation tools such as Spectral[6] (Ferrer et al., 2015). In EDA, LoB teams define event flows and manage Kafka resources through version-controlled YAML[7] specs, with automated validation ensuring compliance. In MFT, business users can configure transfers and security policies using GitOps[8] workflows, with automation orchestrating deployment, error management, and compliance (Folds & McDermott, 2019; Hyvönen et al., 2018; Krishnan et al., 2022).

Automation is a foundational pillar in the C4E model, enabling organizations to reconcile agility and governance at scale. In decentralized environments, where responsibility for integration shifts from centralized IT teams to individual Lines of Business (LoB), manual processes become a bottleneck and a source of inconsistency. Automation addresses these challenges by embedding organizational knowledge, validation rules, and best practices directly into repeatable pipelines and workflows.

Within EDAs, automation orchestrates the entire lifecycle of event resources: from declarative specification in YAML, to validation via tools like Spectral, to automated provisioning and documentation in systems such as EventCatalog[9]. This approach ensures that all event flows adhere to established conventions, are auditable, and are kept up to date with minimal manual intervention. As a result, LoB teams are empowered to innovate and respond to business needs swiftly, without risking drift from governance policies or introducing operational errors.

Equally significant is the role of automation in MFT scenarios. As integration patterns increasingly encompass both real-time and batch-oriented workflows, automated pipelines become essential for managing the complexity and diversity of file exchanges. In a C4E context, automation allows LoB teams to define file transfer jobs—including endpoints, schedules, naming policies, and error-handling logic—through standardized, version-controlled YAML specifications. CI/CD and GitOps pipelines validate these configurations, provision secure credentials via HashiCorp Vault[10], and deploy jobs to the MFT platform in a controlled, auditable manner. Automated monitoring and alerting (e.g., via Grafana[11] and CA Workload Automation[12] known as CAWA) further enhance operational resilience and compliance (Hyvönen et al., 2018; Krishnan et al., 2022).

Theoretical and empirical studies emphasize that automation is not merely a convenience, but a strategic enabler of decentralized IT governance. It enforces consistency, accelerates delivery,

[6] https://meta.stoplight.io/docs/spectral/ (last accessed on Jun, 2025)
[7] https://yaml.org/ (last accessed on Jun, 2025)
[8] https://www.gitops.tech/ (last accessed on Jun, 2025)
[9] https://www.eventcatalog.dev/ (last accessed on Jun, 2025)
[10] https://www.vaultproject.io/ (last accessed on Jun, 2025)
[11] https://grafana.com/ (last accessed on Jun, 2025)
[12] https://www.broadcom.com/products/mainframe/workload-automation (last accessed on Jun, 2025)

reduces cognitive and operational load, and allows organizations to scale integration efforts without linear increases in human resources (Breunig et al., 2020; Bruns & Dunkel, 2014; Ferrer et al., 2015; Hyvönen et al., 2018; Krishnan et al., 2022). In both EDA and MFT domains, automation transforms integration from an artisanal, error-prone process into an industrialized, self-service capability—one that aligns with C4E's vision of democratized, compliant, and business-driven IT (Chukhajyan, 2016; Folds & McDermott, 2019; Vangala et al., 2022).

Ultimately, embedding automation at the heart of resource management is what allows C4E models to deliver on their promise: empowering business teams while safeguarding the standards, security, and reliability that enterprises require.

For EDA, agility is enabled by allowing Line of Business (LoB) teams to rapidly define and deploy new event flows using standardized YAML specifications and automated CI/CD pipelines. This self-service model accelerates onboarding, reduces dependency on central teams, and supports faster adaptation to business requirements. At the same time, governance is maintained through rule-based validation (e.g., Spectral), automated documentation in EventCatalog, and strict enforcement of naming conventions and metadata policies. All changes are auditable and reversible, promoting transparency and compliance.

In the MFT context, agility is achieved by decentralizing the creation and management of file transfer jobs. Business teams can specify endpoints, schedules, file naming policies, and recovery procedures declaratively, leveraging GitOps workflows for rapid deployment and change management. Governance is embedded in the process: all specifications are validated before deployment, secrets are centrally managed via HashiCorp Vault, and robust audit trails are generated automatically (Hyvönen et al., 2018; Krishnan et al., 2022). Production monitoring via Grafana and real-time alerting with CAWA ensure operational integrity and incident response.

Both domains benefit from:

- **Automated compliance**: Organizational standards are enforced through validation pipelines and complete documentation, reducing manual oversight.
- **Version control and traceability**: All changes are tracked in Git repositories, supporting collaboration, rollback, and forensic analysis.
- **Scalability**: The frameworks are designed to accommodate growing integration demands and diverse use cases, without sacrificing control.
- **Security**: Secrets management and role-based access controls mitigate the risk of data breaches and unauthorized operations.

Nonetheless, maintaining this balance requires continual refinement of validation rules, clear definition of roles and responsibilities, and ongoing investment in training and tooling. By integrating agility and governance as core architectural principles—rather than as competing priorities—this dissertation demonstrates a sustainable path for democratized, compliant, and resilient integration in enterprise environments.

2.5 Frameworks and Best Practices

The development of the self-service integration framework is underpinned by two distinct but complementary technological stacks (see table 2.1), tailored to support both EDA and MFT workflows. Each stack is designed to address the unique challenges and requirements of its domain, while together enabling a unified, democratized approach to integration within the organization.

For the EDA component, the core of the solution is **Apache Kafka**, a robust event streaming platform recognized for its scalability, high throughput, and reliability. Kafka enables asynchronous communication and system decoupling, and its widespread adoption within the organization ensures operational continuity and ease of onboarding.

To maintain transparency and governance across the event ecosystem, **Event Portal** serves as a centralized documentation portal, capturing event specifications, topics, schemas, and ownership details. This enables organizational standardization and provides teams with immediate insight into available integrations and dependencies across business units.

All integration artefacts—including events, topics, producers, consumers, and service roles—are modeled using **YAML**. YAML's human-readable and flexible syntax makes it ideally suited for configuration-as-code, promoting collaborative management and version control of integration resources via Git repositories.

Spectral is incorporated within the CI/CD pipelines to enforce compliance with organizational standards. Every pull request triggers Spectral's automated validation of YAML files, verifying naming conventions, mandatory metadata, schema references, and structural integrity. This process acts as a critical quality gate, reducing the risk of misconfiguration and ensuring consistent resource definitions.

The automation layer supporting EDA is developed in **Python**[13], chosen for its powerful library ecosystem and ease of integration with external APIs. Python scripts are responsible for parsing YAML configurations, provisioning Kafka topics and Access Control Lists (ACLs), and synchronizing documentation with Event Portal. This approach ensures flexibility, extensibility, and reliable error handling within the automation framework.

While **Jenkins**[14] is operated by a dedicated DevOps team (and not directly modified in this dissertation), it orchestrates the execution of all automation scripts for the EDA pipeline. Jenkins triggers validation, provisioning, and documentation updates on pull request events, guaranteeing that all infrastructure changes are systematically validated and deployed only after passing stringent quality checks.

In summary, the EDA technology stack delivers a resilient, transparent, and self-service foundation for event integration, balancing organizational agility with robust governance and compliance at every step of the resource lifecycle.

Turning to the MFT solution, the technological foundation is equally robust, yet adapted to the specific needs of secure, auditable, and automated file movement across diverse systems.

[13] https://www.python.org/ (last accessed on Jun, 2025)
[14] https://www.jenkins.io/ (last accessed on Jun, 2025)

As with EDA, YAML is used as the primary specification language, defining the characteristics of MFT flows—including endpoints, authentication methods, error and post-processing logic, and ownership metadata. These YAML files serve as the single source of truth for all MFT configurations.

To guarantee compliance and governance, Spectral is once again employed to validate all MFT specifications against a set of custom organizational rules before deployment. This ensures that only well-formed and policy-compliant configurations are promoted to pre-production and production.

The adoption of **GitOps** principles using **GitHub Actions**[15] underpins the entire MFT deployment process. All configuration changes are managed through Git repositories, enabling declarative, version-controlled infrastructure, complete audit trails, and automated rollbacks in case of errors. This approach provides both traceability and reproducibility, supporting continuous improvement and compliance.

Security is a central concern in file transfer scenarios; **HashiCorp Vault** is used for the secure storage and management of secrets—such as credentials and API tokens—required for authenticating MFT flows with external endpoints. Vault's centralized approach enables strong access controls and auditability, significantly reducing the risk of credential leakage or misuse.

Grafana powers the monitoring dashboards for the MFT platform, providing real-time visibility into platform health, throughput, and operational status. This proactive monitoring is complemented by **CA Workload Automation**, which handles automated alerting and incident management for production environments, ensuring that operational teams are immediately notified of and able to respond to any anomalies or failures in MFT operations.

At the center of the solution is the **MFT Platform** itself. This orchestrator is responsible for executing secure, reliable, and governed file exchanges between systems, supporting a wide range of protocols—including FTP and SFTP—as well as modern cloud storage backends such as Azure Blob Storage and Amazon S3. The platform enforces connection management, advanced error recovery, auditing, scheduling, pooling, and detailed file transfer monitoring.

Together, these two technology stacks—EDA and MFT—embody a comprehensive, future-proof strategy for integration, enabling the organization to meet business demands with speed, security, and confidence, all within a strong governance framework—an essential requirement for aligning technological architecture with business objectives, as emphasised by Ross et al., 2006.

This technological landscape ensures that both EDA and MFT workflows benefit from automation, strong governance, auditability, and operational resilience, each leveraging a best-of-breed stack suited to their unique integration challenges.

In addition to the selected technologies, robust frameworks and methodologies guide the development and implementation of both EDA and MFT solutions, ensuring consistency, scalability, and efficiency across the integration landscape.

DevOps practices are central to this approach, fostering collaboration between development and operations teams. By emphasizing automation, Infrastructure as Code (IaC), and continuous

[15] https://github.com/features/actions (last accessed on Jun, 2025)

Table 2.1: Summary of Technologies by Integration Domain

Technology	EDA (Event-Driven Architecture)	MFT (Managed File Transfer)
Apache Kafka	✓	
EventCatalog	✓	
YAML	✓	✓
Spectral	✓	✓
Python	✓	
Jenkins	✓[a]	
GitOps		✓
HashiCorp Vault		✓
Grafana		✓
CA Workload Automation (CAWA)		✓[a]
MFT Platform		✓

[a]Implemented by another team (not covered in this dissertation).

feedback, DevOps minimizes manual errors, accelerates delivery cycles, and improves the overall reliability of integration workflows.

For EDA, **CI/CD (Continuous Integration/Continuous Deployment)** pipelines automate the validation, testing, and deployment of YAML specifications for Kafka resources and Event Portal updates. These pipelines leverage Spectral for rule-based validation, guaranteeing compliance with naming conventions, required fields, and documentation standards before resources are promoted to production.

For MFT, the framework incorporates **GitOps principles**, where all file transfer configurations are managed declaratively via Git repositories. Each change to an MFT YAML specification triggers a GitHub Actions workflow that validates the configuration (again using Spectral), applies business and security policies, and deploys the transfer job to the MFT platform. This version-controlled, auditable process enables safe rollbacks, traceability, and collaborative change management across teams.

Secrets management is ensured by integrating with HashiCorp Vault, which securely stores and provides credentials for automated pipelines without exposing sensitive data in code or repositories. Access to secrets is tightly controlled on a per-team basis, ensuring that each team can only access its own credentials and resources, with no visibility into the secrets of other teams. This approach significantly reduces the risk of unauthorized access and reinforces organizational security best practices.

Monitoring and notification frameworks complete the methodology: Grafana dashboards provide real-time visibility into platform health and MFT has it's own transfer/job status and monitoring, while CAWA delivers production-grade alerting and incident management for MFT flows, ensuring rapid response to operational issues.

By integrating DevOps, GitOps, IaC, automated validation, and comprehensive monitoring, this framework ensures that both EDA and MFT processes are robust, compliant, and scalable—empowering

teams with the autonomy of self-service while maintaining organizational governance and operational excellence.

2.6 Key Findings

The selected studies provide diverse perspectives on the design and implementation of EDAs, particularly in terms of scalability, decoupling, and responsiveness within distributed systems. Across the literature, EDA is widely recognized for enabling asynchronous communication, improving system modularity, and supporting real-time data processing in heterogeneous environments (AboElHassan & Yacout, 2023; Khriji et al., 2022; Lopes & Oliveira, 2015).

A recurring theme is the role of automation as a catalyst for operational efficiency. Several authors emphasize the use of automated pipelines, validation mechanisms, and deployment workflows to reduce manual dependencies and ensure compliance with organizational standards (Estruch & Álvaro, 2012; Ferrer et al., 2015; Pan et al., 2024). These mechanisms are especially valuable when transitioning to a self-service integration model, where LoB teams are empowered to manage their own data flows within predefined governance constraints (Folds & McDermott, 2019; Krishnan et al., 2022).

In the context of governance, the C4E model is presented as a modern alternative to the traditional CoE. The literature suggests that C4E promotes agility and decentralization by shifting integration responsibilities closer to business teams (Breunig et al., 2020; Bruns & Dunkel, 2014; Vangala et al., 2022). However, it also highlights challenges in maintaining consistency, quality, and security when authority is distributed. This tension between autonomy and control is a key concern addressed by various authors through proposals for layered governance, embedded automation, and centralized oversight of decentralized processes (Romero et al., 2022).

Kafka emerges as a foundational technology in many of the reviewed implementations, underlining its suitability for building high-throughput, low-latency data pipelines. Beyond its role in event streaming, Kafka is frequently positioned as the backbone of modern data platforms, enabling event sourcing, real-time analytics, and scalable microservice communication (Bhat & H, 2024; Khriji et al., 2022; Molina et al., 2018).

While EDAs dominate real-time data exchange, many enterprises still rely heavily on file-based integration for structured, batch-oriented data transfer. This is particularly relevant in industries dealing with legacy systems, regulatory reporting, or B2B data exchange, where files remain the standard mechanism for interoperability.

Several studies from the reviewed literature reinforce the enduring relevance of MFT systems in such contexts (Chukhajyan, 2016; Hyvönen et al., 2018; Krishnan et al., 2022). These systems provide structured, secure, and auditable ways to exchange files across applications, partners, or organizational boundaries. MFT platforms typically offer automation features like scheduling, encryption, endpoint validation, and monitoring capabilities to ensure compliance and reduce operational risk.

Hyvönen et al. (2018) highlight the integration of automated file flows in governmental IT services, emphasizing the role of MFT in standardizing file formats and reducing manual intervention. Similarly, Krishnan et al. (2022) notes that MFT enables robotics-driven manufacturing systems to handle structured file communication in industrial IoT networks.

Moreover, the transition to self-service MFT models is gaining traction as organizations seek to decentralize integration responsibilities. This mirrors the shift seen in event-driven platforms. Chukhajyan (2016) propose automated validation and configuration workflows that allow non-specialized teams to initiate and manage file transfers securely, supporting the C4E governance approach.

While traditional MFT models rely on centralized configuration and deployment, newer frameworks emphasize modular, user-driven orchestration to reduce dependency on IT support teams. Folds and McDermott (2019) underscores the importance of designing scalable file systems where LoB teams can independently define rules for file generation, routing, and archival within approved governance boundaries.

These findings confirm that file-based integration—far from obsolete—remains a vital complement to EDA, particularly when equipped with self-service and automation capabilities. Both integration modes benefit from similar governance structures and enablement principles, reinforcing the feasibility of a unified, democratized integration strategy.

Some studies go further by evaluating the organizational impact of adopting EDA and decentralized governance. These works suggest that the success of such transitions depends not only on technical enablers but also on cultural readiness, the maturity of supporting tools, and clear alignment between IT and business objectives.

Chapter 3

Problem and Solution Approach

This chapter presents the core problem addressed by this dissertation and outlines the methodological approach adopted to solve it. It begins by analyzing the main challenges that arise when transitioning from a centralized CoE to a decentralized C4E in enterprise integration contexts—particularly focusing on the organizational, technical, and operational risks introduced by such a shift. Following this, the chapter introduces the DSR methodology as the guiding framework for artefact development and evaluation. It then details how DSR was specifically applied to this research, describing the process of identifying requirements, designing and implementing self-service integration artefacts, and evaluating their impact in a real-world business environment. This chapter establishes the link between the identified integration challenges and the innovative solutions presented in the dissertation, laying the foundation for the results and discussion in subsequent chapters.

3.1 Problem Definition

One of the main challenges growing organizations face is integrating heterogeneous technical systems, which often rely on different technologies, communication protocols, and data formats. Therefore, system integration can be broadly defined as connecting various applications, systems, or data streams to enable them to interoperate efficiently and effectively. This process often includes managing the data flow, orchestrating system behavior, and ensuring operational consistency across different technological components.

Traditionally, the increasing complexity of enterprise IT landscapes has led to the formation of highly specialized integration teams, often centralized under a CoE (Fishman, 2022) governance model. In this model, a single team is responsible for defining standards, implementing integration logic, and maintaining critical systems and services. Although this ensures high levels of standardization and control, it also introduces organizational bottlenecks, especially in dynamic environments where rapid change and responsiveness are crucial.

Many organizations are adopting a decentralized approach, the C4E (Fishman, 2022) to mitigate these inefficiencies. In the C4E model, responsibility is distributed to individual teams,

commonly called Lines of Business (LoBs) (Ovington, 2023)—equipped with frameworks, tools, and governance guidelines to autonomously develop and manage technical resources. This model preserves a certain level of oversight while promoting autonomy and agility across the organization.

This dissertation focuses on implementing democratized integration practices in two distinct yet foundational enterprise integration patterns: EDA (Seetharamugn, 2023) and file transfers.

On one hand, event-driven integration enables asynchronous communication between systems by publishing and consuming events. In a typical enterprise scenario, a new event flow involves coordination across multiple teams—application developers, integration specialists, and platform operators—which results in long delivery times due to inter-team dependencies and sequential workflows. This dissertation proposes a model where LoB teams can define event flows themselves, submitting simplified, structured specifications that are processed automatically. This approach reduces time-to-market and alleviates pressure on the central integration team.

On the other hand, file-based integration remains a widely used paradigm, especially in industries with legacy systems or batch data exchange requirements. Traditionally, this type of integration has been handled through SOA (Papazoglou & van den Heuvel, 2007) frameworks or custom scripts, managed centrally by the Integration team. However, these approaches are rigid, slow to adapt, and place a heavy operational burden on specialized resources. This work also presents a self-service approach for file transfers, the MFT, wherein business teams can request, configure, and manage file transfers independently through automated interfaces with built-in validation and governance mechanisms. This modernized solution will replace the company's legacy SOA-based file exchange model.

This work demonstrates how democratizing integration capabilities through decentralizing event-driven and file-based processes can increase a large enterprise's efficiency, scalability, and responsiveness. The implementation is contextualized in a real-world organization; therefore, it can serve as a replicable model for other companies with similar technological and operational constraints.

In traditional CoE (Fishman, 2022) models, integration teams are often the sole entities responsible for developing, deploying, and maintaining system integration solutions. While this ensures standardization and high technical quality, it often leads to organizational bottlenecks. In rapidly evolving environments, where business agility and time-to-market are critical, this centralization hampers innovation and slows down the delivery of business value.

These inefficiencies are particularly evident in scenarios such as event-driven development and file-based integrations. Depending on centralized resources, business teams must wait in line for their requests to be prioritized, specified, and implemented. The delivery process often operates under a traditional project model, managed with methodologies like Agile or Waterfall (Laoyan, 2024), which adds further delays.

By contrast, the C4E (Fishman, 2022) model encourages the delegation of integration responsibilities to LoB (Ovington, 2023) teams, who can act more quickly and autonomously. Teams can handle their own event-based or file-based integration needs through automation, templated

validation, and standardized self-service frameworks. The result is reduced development and deployment time, improved scalability of integration services, and better alignment between IT capabilities and business objectives.

While transitioning to a C4E (Fishman, 2022) model brings substantial benefits, it also presents new challenges. One of the main concerns is maintaining delivery quality when responsibilities are transferred from highly specialized teams to business-oriented teams with varying levels of technical proficiency. Without proper governance, automated validation, and documentation, this shift can introduce security, consistency, and maintainability risks.

In the context of EDAs, the central integration team often becomes the bottleneck, as they are responsible not only for managing the event platform (e.g., Apache Kafka) but also for implementing and provisioning new event flows on behalf of LoB (Ovington, 2023) teams. This leads to inefficiencies and distracts the central team from its core mission: maintaining the platform's stability and governance.

Similarly, in the file-based integration domain, legacy SOA-based solutions have proven to be inflexible and time-consuming. Each new file transfer flow requires custom configuration and coordination, limiting scalability and increasing operational load. This outdated model is incompatible with modern business demands for responsiveness and adaptability.

This dissertation addresses both problems by proposing a self-service model that allows LoB (Ovington, 2023) teams to autonomously define and manage event flows and file transfers within a framework that includes validation, automation, and centralized governance. This approach ensures that integration processes are scalable and reliable, even in a decentralized operating model.

This dissertation's research question, introduced in Section 1.3, assesses the organizational, technical, and operational outcomes of democratizing integration processes. It investigates explicitly whether enabling business teams to manage events independently—driven and file-based integration resources can deliver tangible improvements in agility and delivery speed without sacrificing quality and governance.

3.2 Challenges in Democratized Integration

The transition from a centralized CoE to a decentralized C4E model presents a set of multi-faceted challenges that span both technical and organizational domains (Bruns & Dunkel, 2014). While the C4E approach promises increased agility and empowerment for business teams, it also exposes potential risks and operational complexities, especially in heterogeneous integration scenarios involving both EDA and MFT.

- **Governance vs. Agility**: Decentralizing resource management can threaten the uniformity and quality standards that were previously safeguarded by centralized teams. Balancing the need for rapid innovation with adherence to security, compliance, and data governance policies remains a persistent challenge (Peng et al., 2023).

- **Specialization Gaps**: As LoB teams assume greater ownership of integration flows, gaps in technical expertise and architectural knowledge may arise. This is particularly acute in areas such as advanced Kafka topic configuration, MFT security policies, or error-handling strategies, requiring investment in robust frameworks, training, and user-friendly automation tools (Breunig et al., 2020).

- **Scalability of Automation**: Designing and maintaining automation pipelines that can accommodate the diverse needs of multiple LoB teams—each with unique workflows, protocols, and compliance requirements—demands careful planning. Automation must be flexible yet governed, scalable yet manageable, for both real-time (EDA) and batch-oriented (MFT) processes (Ferrer et al., 2015).

- **Documentation and Traceability**: In a decentralized model, ensuring that every integration flow—be it an event stream or a file transfer job—is fully documented, version-controlled, and auditable is critical to long-term governance and operational resilience.

- **Security and Secret Management**: Exposing configuration and deployment tasks to a broader set of users increases the risk of misconfiguration or credential leakage, necessitating strong access controls, automated secrets management (e.g., HashiCorp Vault), and clear policies for incident response.

- **Platform Complexity**: The adoption of platforms such as Kafka (for EDA) and advanced MFT solutions introduces their own operational complexities. For example, Kafka resource management requires strict validation of events, topics, and ACLs, while MFT flows must enforce naming conventions, atomicity, and secure transmission protocols. Without careful automation and governance, these complexities can lead to misconfigurations, system vulnerabilities, or failed integrations.

Addressing these challenges requires not only robust automation and governance frameworks but also continuous investment in organizational change management, training, and cross-functional collaboration (Forsgren et al., 2018).

The state of the art in enterprise integration is defined by the movement toward decentralization, automation, and self-service, as exemplified by C4E, EDA, and MFT models. Combining robust technology stacks, automation, and governance, organizations can reduce bottlenecks and deliver business value faster. However, the ongoing balance between agility and compliance, and investment in people and processes, remains crucial for long-term success.

3.3 Design Science Research Methodology (DSR)

DSR is a rigorous methodology in information systems and engineering, structured specifically for the purposeful creation and evaluation of artefacts that solve relevant, complex, and real-world organizational problems (Hevner, 2007; Peffers et al., 2007). Rather than focusing solely on

explaining or predicting phenomena, DSR seeks to advance knowledge through the iterative design and assessment of constructs such as frameworks, processes, models, or technological solutions. The aim is to bridge the gap between theory and practice, offering artefacts that are simultaneously innovative, useful, and theoretically grounded.

A defining feature of DSR is its framework comprising three interrelated cycles (Figure 3.1):

- **Relevance Cycle:** This cycle connects the research to the environment and application domain—such as people, organizational systems, and technical challenges—ensuring that the artefact addresses genuine, high-impact needs. It focuses on capturing requirements, identifying problems and opportunities, and validating artefact usefulness through field testing.
- **Design Cycle:** At the core of the methodology, the Design Cycle involves the iterative construction (building) and assessment (evaluation) of the artefact itself. Feedback from evaluations is continuously fed back into the design process, supporting refinement and adaptation until both utility and scientific contribution are achieved. The cycle ensures that the artefact is not only theoretically sound but also effective in practical application.
- **Rigor Cycle:** This cycle ensures that the research is informed by and contributes to the scientific knowledge base. It draws on foundations such as established theories, empirical evidence, best practices, and prior artefacts, and guarantees that new artefacts and insights are well grounded. It also feeds validated results back into the knowledge base, advancing both theory and practice.

The DSR framework, therefore, integrates the demands of the organizational environment with the academic pursuit of rigor, cycling iteratively between building, evaluating, and grounding artefacts, while remaining firmly connected to both real-world requirements and the body of scientific knowledge (Hevner, 2007). This methodological structure is particularly well suited to the development and validation of digital integration solutions in complex enterprise contexts.

3.4 DSR Applied to the Problem

The DSR methodology (Figure 3.2) was systematically applied to address the central research question of this dissertation: How does the transition from a centralized CoE to a decentralized C4E model—through the implementation of automated pipelines and self-service frameworks for event-driven and file-based integration—impact efficiency, scalability, and governance in large-scale enterprise environments?

Relevance Cycle: The Relevance Cycle established a strong link between the research objectives and the real-world environment at Sonae MC, specifically within MC Digital. This began with a comprehensive diagnosis of the organizational context: a large multinational retailer facing increasing pressure for digital transformation and integration between heterogeneous systems.

A critical component of the Relevance Cycle was a comprehensive literature review A, which informed and validated the research direction. This review explored recent advancements and

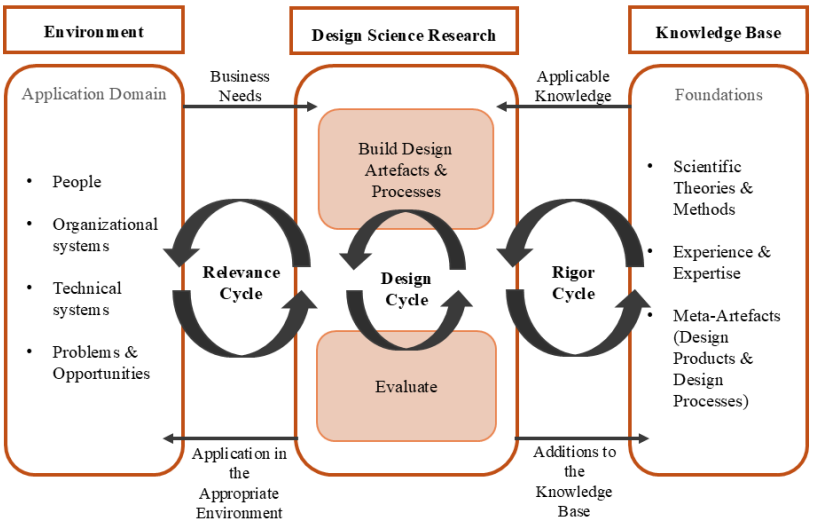


Figure 3.1: Design Science Research (DSR) Framework (Hevner, 2007).

best practices in decentralized IT governance, automation, EDAs, and MFT solutions. It provided essential theoretical foundations and benchmarks by identifying how leading organizations overcome similar challenges through the adoption of C4E models, self-service frameworks, and automation pipelines. The literature review also helped to refine the problem statement, justify the choice of artefacts, and identify performance and governance metrics for later evaluation.

Key challenges were identified with IT and business teams, and an analysis of existing operational bottlenecks. These included delays and inefficiencies in system integration due to the centralized CoE structure and the absence of scalable, self-service tools for Lines of Business (LoB) to independently create and manage their own integrations. Concrete objectives were therefore defined, such as designing automated self-service frameworks for EDA and MFT, enforcing validation and access control, and measuring the organizational impact of a transition to C4E. Benchmarking against the legacy SOA/Connectivity solution and collecting quantitative data (e.g., onboarding time, transfer speeds, number of user tickets) ensured that the project targeted high-impact needs.

Design Cycle: The heart of the methodology resided in the iterative Design Cycle, where artefacts were both constructed and evaluated in real operational contexts. Drawing from the requirements surfaced in the relevance cycle, two core solutions were designed: an EDA self-service automation framework and a MFT self-service platform. Artefacts were developed using a combination of Python automation, YAML/JSON declarative specifications, CI/CD pipelines (Jenkins, GitHub Actions), and validation engines such as Spectral. Each artefact was iteratively refined through cycles of prototyping, implementation in test environments, and stakeholder feedback. Quantitative analysis (onboarding lead times, reduction in manual intervention, transfer throughput) and qualitative analysis (personas, user satisfaction, governance adherence) provided multidi-

mensional insights for each iteration. Evaluation activities also included the creation of personas to capture changes in developer and business user experience before and after the implementation of the frameworks.

Rigor Cycle: The Rigor Cycle ensured that all developments were grounded in academic theory, best practices, and industry models. This involved an extensive literature review on EDAs, self-service integration, IT governance, automation, and DevOps/GitOps principles, as detailed in Chapter 2. Established models—such as the distinction between CoE and C4E, and frameworks for continuous integration, validation, and secrets management—were incorporated to anchor artefact design in recognized foundations. Conversely, the results, insights, and improvements from the design and evaluation of the artefacts were formally documented and disseminated, enriching both the internal knowledge base at MC Digital and contributing new practical evidence to the academic literature.

Data Collection and Analysis: Data collection combines several approaches, such as, analysis of pipeline logs (to measure onboarding speed, failure rates, and user adoption), documentation review, and direct interviews with developers and business analysts. A comparative analysis of KPIs—such as lead time, error rates, and number of manual interventions—before and after the implementation of the self-service frameworks provided robust evidence of organizational change. The inclusion of personas and feedback cycles further enabled a nuanced understanding of how the transition to C4E empowered teams, reduced bottlenecks, and drove a cultural shift towards proactive enablement and continuous improvement. To capture these qualitative insights, I conducted targeted interviews and distributed structured questionnaires to a diverse group of stakeholders actively engaged in the new integration workflows. The narratives presented as personas throughout this dissertation are, therefore, based on real feedback and experiences collected during this research. For reasons of confidentiality and in compliance with data protection guidelines, the actual names and identities of participants have been replaced by fictional names. These fictitious personas serve to illustrate genuine user journeys and perspectives, ensuring both the authenticity of the analysis and the anonymity of respondents.

Synthesis: By cycling iteratively through these phases (Figure 3.2), the research produced two rigorously evaluated, operational self-service artefacts (EDA and MFT), established new governance models, and generated actionable, generalizable knowledge for future enterprise integration projects. This approach ensured both immediate business impact and the creation of transferable scientific value.

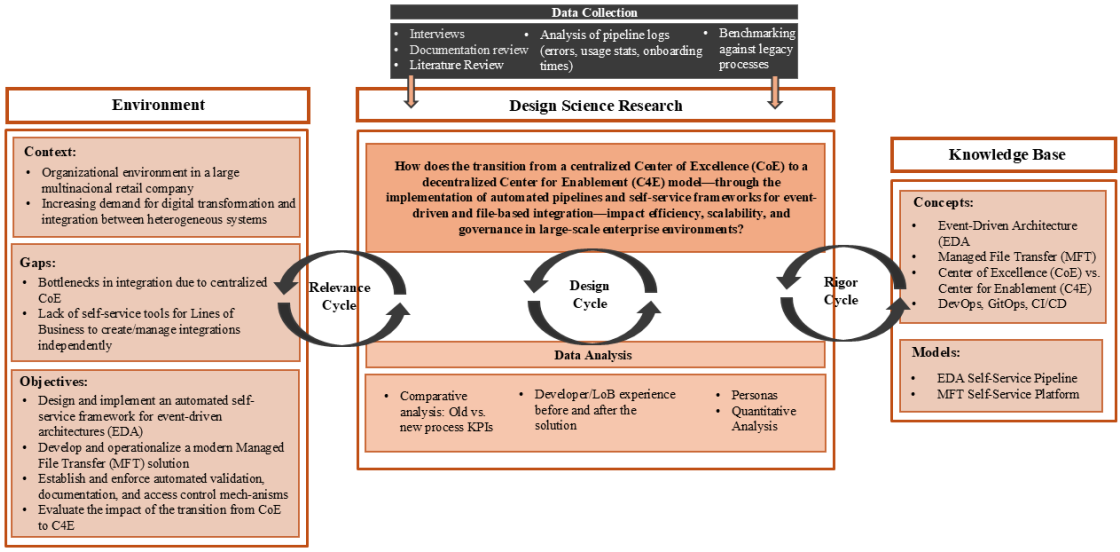


Figure 3.2: DSR framework applied to the problem.

Chapter 4

Development and Implementation

The core problem addressed in this dissertation is the bottleneck and lack of agility resulting from CoE integration models. In response, this research applies DSR by developing two innovative, self-service artefacts. An automated EDA framework, leveraging YAML-based specifications, Python automation scripts, and integration with Kafka and EventCatalog. A MFT pipeline, based on declarative YAML, GitOps automation, secret management with HashiCorp Vault, and integration with MFT platforms. Each artefact is designed, implemented, demonstrated, and evaluated in the real-world context of MC Digital/Sonae MC, empowering LoB teams to manage their own integrations with full compliance and minimal manual intervention.

4.1 Event-Driven Architecture (EDA) Automation

One of the main goals of this project is to empower both technical and non-technical team members to define integration artefacts. The use of human-readable YAML and guided templates means that users can easily create or update events and services. The EDA automation solution is architected as a modular suite of Python scripts that orchestrate the lifecycle of Kafka resources and event/service documentation. The system operates directly on the Kafka cluster to create topics and ACLs and integrates seamlessly with the Event Portal for documentation. All resources are declared through YAML and JSON specifications, versioned in a Git repository, and subject to automated validation and approval workflows. The specification process is documented step-by-step:

1. Check existing events in the Event Portal.
2. If a new event is required, create a YAML event specification, a JSON schema, and a sample payload.
3. For service integration, either update an existing service YAML or create a new one.
4. Publish your branch remotely to create the Kafka topics and ACLs in pre-production environment.
5. Submit a pull request for review and validation.

6. Upon approval, the automation pipeline will create the Kafka topics and ACLs in production environment and create a Pull Request in another repository to update documentation.

7. Retrieve or request user credentials as required for Kafka access.

Naming conventions are enforced programmatically, reducing errors and ensuring all artefacts are discoverable and searchable.

All event and service definitions are specified in YAML, schema payload and payload examples are JSON files:

- `integration-event-specs/events/`: Event specifications (YAML), see Figure 4.1.

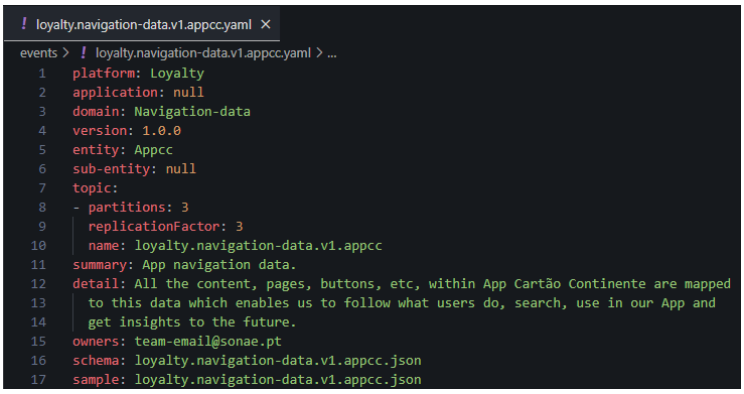


Figure 4.1: Event specification example.

- `integration-event-specs/services/`: Service specifications (YAML), see Figure 4.2.

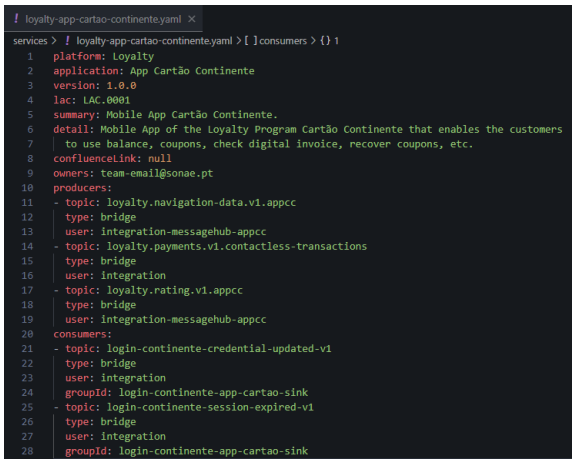


Figure 4.2: Service specification example.

- `integration-event-specs/schemas/`: Event payload schemas (JSON), see Figure 4.3.

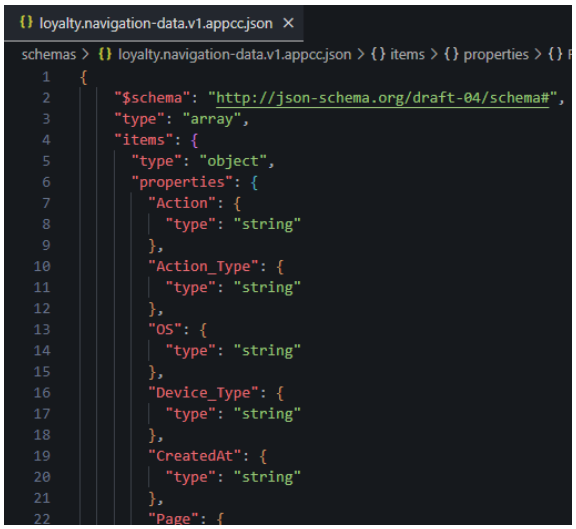


Figure 4.3: Event schema example.

- `integration-event-specs/samples/`: Example event payloads (JSON), see Figure 4.4.

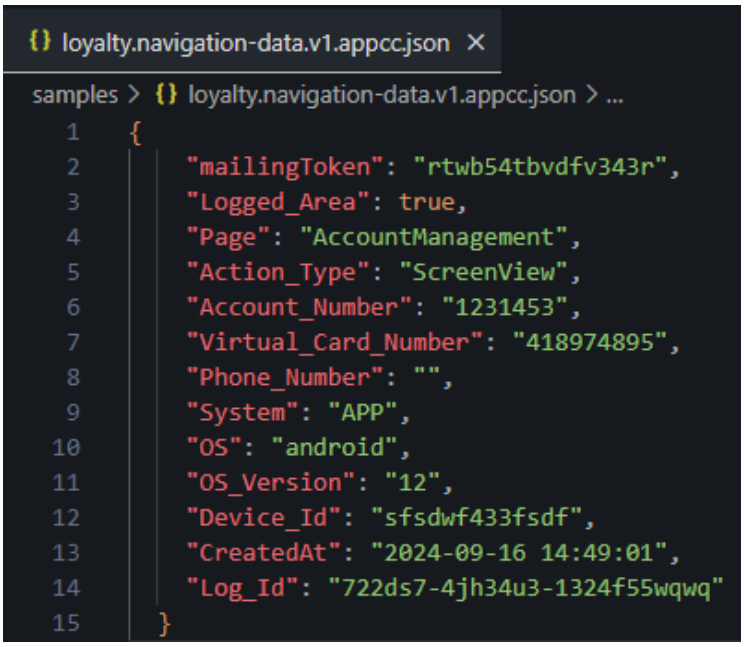


Figure 4.4: Event payload example.

This structure promotes separation of concerns, enables self-service, and ensures a declarative and auditable model for integration development.

Technical Architecture and Modularity

The EDA automation framework is architected as a modular suite of Python scripts, each responsible for a distinct phase of the integration artefact lifecycle:

- **Specification and Validation:** Users create or update YAML and JSON files, which are versioned in a central Git repository. Upon each commit or pull request, automated validation is triggered.
- **Resource Creation:** Python scripts interact directly with the Kafka administration APIs to create or update topics and configure ACLs as specified in the YAML files.
- **Documentation Synchronization:** Event and service metadata are automatically compiled and submitted as pull requests to the Event Portal documentation repository, ensuring that organizational knowledge is always up to date.

This modular approach supports future extensibility, such as integration with additional streaming platforms or documentation tools.

Pipeline Validations

Validation of pull requests (PRs) is a central feature, ensuring only compliant and complete configurations are merged and applied. The `pipeline_validations.py` script checks for the existence and correctness of all necessary files. Spectral or custom validation tools can be integrated for enforcing organizational naming conventions and schema consistency.

The validation phase is crucial to guarantee compliance and quality. The `pipeline_validations.py` script, together with Spectral, verifies:

- All required files exist and are syntactically correct.
- YAML files adhere to organizational naming conventions.
- JSON schemas are valid and compatible with their respective samples.
- All owners and contact details are correctly specified.

Validation is triggered automatically on each pull request, providing immediate feedback and blocking non-compliant artefacts from being merged.

Script Orchestration

The automation framework is divided into two principal groups of scripts:

1. **Kafka Management Scripts:**
    - `pipeline_create_topics.py`: Automates the creation of Kafka topics, supporting both live and dry-run modes for safe validation.
    - `pipeline_set_acls.py`: Automates the setup of Access Control Lists for topics, groups, and sinks.
2. **Event Portal Pipeline Scripts:**
    - `pipeline.py`: Main orchestrator, responsible for coordinating the end-to-end process, including execution order, logging, and error handling.
    - `pipeline_events.py` and `pipeline_services.py`: Specialized modules for processing event and service specifications, respectively.
    - `pipeline_validations.py`: Responsible for enforcing compliance and completeness before artefact processing.

Each script is designed for clarity, maintainability, and ease of extension.

Step-by-Step Workflow

The specification and deployment process is transparent and auditable:

1. The user checks the Event Portal for existing events, preventing duplication.
2. For new events, the user creates a YAML specification, accompanying JSON schema, and a representative payload sample.
3. Service integrations are declared in separate YAML files—either updating existing specifications or creating new ones.
4. The user works in a dedicated feature branch. Upon pushing commits, the automation pipeline validates artefacts and, if compliant, deploys resources to the pre-production (PP) Kafka environment.
5. After successful review, a pull request to the `main` branch triggers validation and deployment in the production (PRD) environment.
6. In parallel, the pipeline updates documentation and manages access credentials as needed.

CI/CD, Branching, and Approval Workflows

The development lifecycle follows modern DevOps practices:

- Feature development and integration tests occur in feature branches (`feature/...`) mapped to the PP environment.
- Pull requests to `main` trigger final validation and deployment to production.
- All pipeline execution steps—including validation, provisioning, and documentation updates—are fully automated and visible via Jenkins and GitHub.
- This trunk-based development model minimizes environment drift and ensures auditability.

The development process (see Figure 4.5) for integration artefacts follows a trunk-based workflow, relying solely on the `main` branch to represent the production (**PRD**) environment. Feature development and integration testing are performed on dedicated feature branches named `feature/...`, which correspond to the pre-production (**PP**) environment. Once a feature branch is validated, a pull request (PR) is created and, upon approval, merged into `main`, thereby promoting the changes to production.

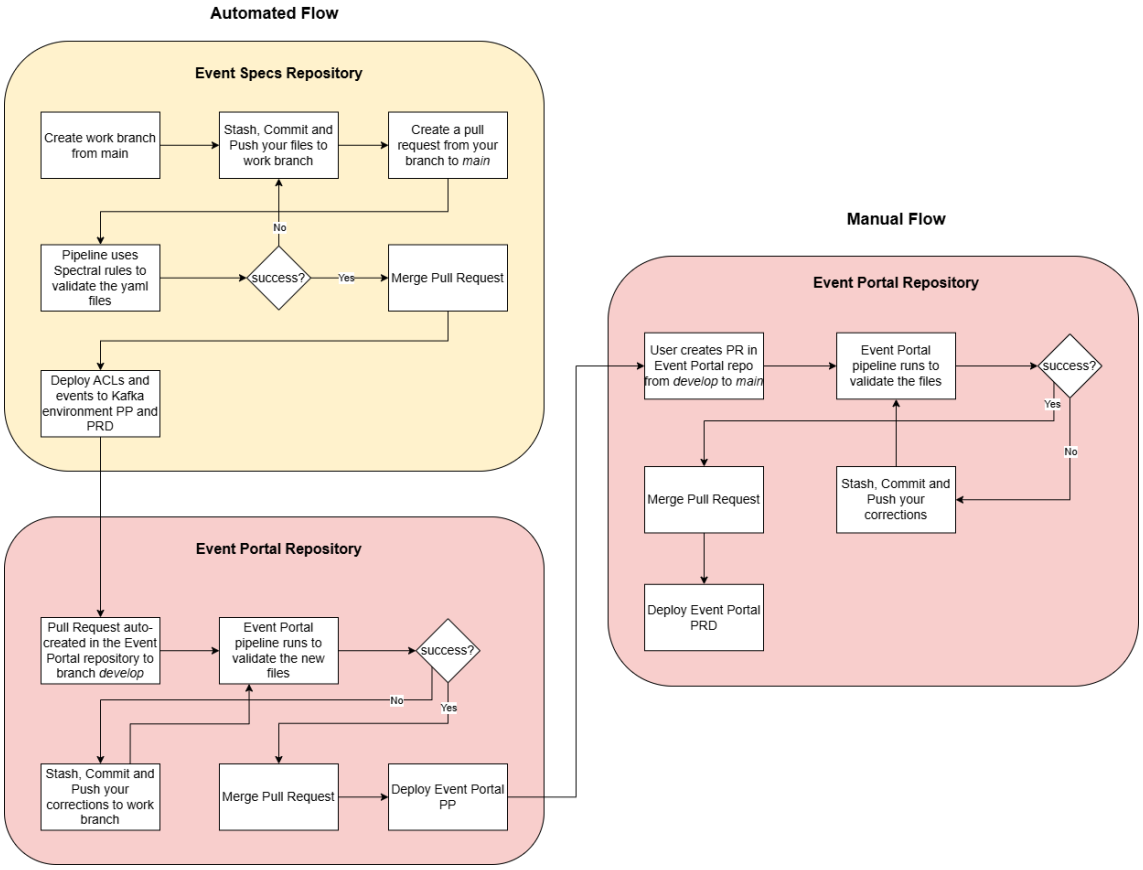


Figure 4.5: EDA Pipeline Workflow.

- All integration artefacts for the PP environment are maintained and tested on the corresponding `feature/...` branch.

- Only artefacts merged into `main` are applied in PRD, ensuring strict separation between development/testing and production.
- The pipeline scripts for topic and ACL creation, as well as event and service processing, are executed within the CI/CD pipeline triggered by PR events.
- This approach minimizes drift between environments and reduces the risk of configuration errors when promoting artefacts to production.

Kafka Resource Provisioning

Resource creation (topics, ACLs) is performed via programmatic interactions with Kafka's administration API. Authentication and connectivity details (brokers, CA, SASL credentials) are securely injected at runtime, following best practices for secret management (e.g., use of environment variables or enterprise vaults). This architecture supports both pre-production and production deployments, with environment segregation enforced at the pipeline level.

EventCatalog Documentation Integration

Automatic documentation is a distinguishing feature of the framework. Every approved integration artefact is documented in the Event Portal, providing up-to-date, discoverable information on available events, producers, consumers, and data schemas (see Figure 4.6). This approach guarantees transparency, simplifies onboarding for new teams, and supports incident response.

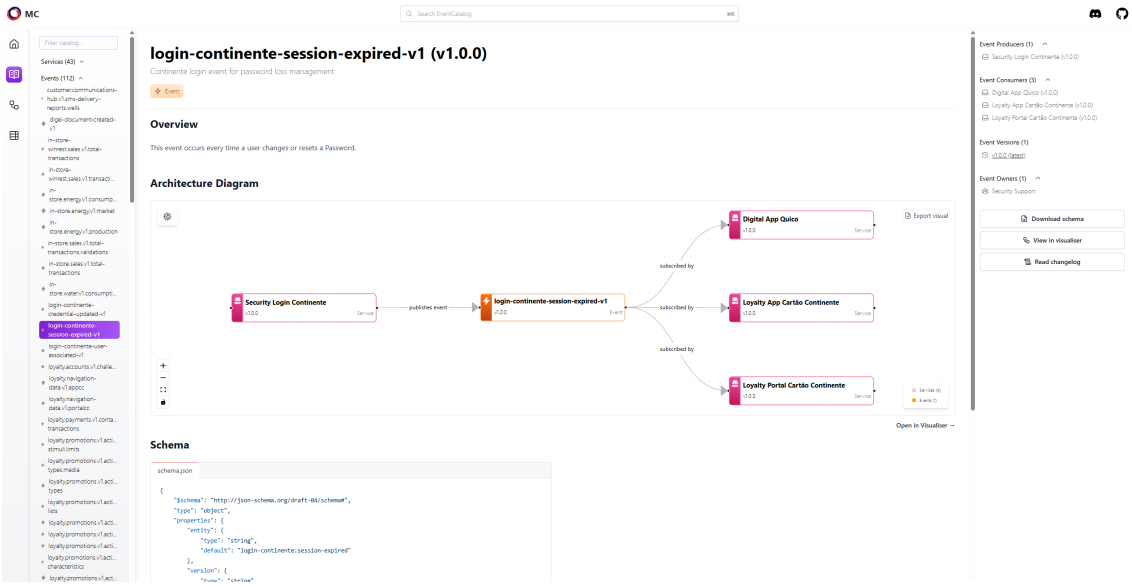


Figure 4.6: Event Portal Example.

Security and Access Control

Security is a foundational concern throughout the artefact's lifecycle:

- No sensitive credentials are hardcoded in scripts or YAML files. Secrets are managed via secure channels and never exposed in logs.
- Kafka topic access is managed strictly via ACLs, ensuring only authorized users/services can produce or consume data.
- User credentials are generated but provisioned through formal approval workflows, preserving segregation of duties and minimizing privilege escalation risk.
- All documentation includes best-practice reminders for secret handling, further supporting organizational security policies.

Monitoring and Error Handling

All execution steps—successes and errors—are logged in Python and made visible through the Jenkins UI. In the event of pipeline failures, relevant error messages are automatically relayed to GitHub, triggering notifications for the responsible user or team. This transparency facilitates rapid debugging and continuous improvement.

Empowerment and Organizational Impact

By lowering the barrier to entry for defining integration artefacts, the EDA automation solution has fundamentally changed the role of LoB teams. Instead of relying on specialized integration engineers, business users can now initiate, document, and deploy new event flows with confidence—knowing that all governance and security checks are embedded in the workflow. This has led to significant improvements in responsiveness, reduction in lead times, and better alignment between IT capabilities and business objectives.

Real-World Example

A representative use case proceeds as follows:

1. A business user, requiring a new sales event integration, consults the Event Portal and creates the necessary YAML specification, JSON schema, and payload sample.
2. The user submits a feature branch; the automation pipeline validates the artefacts and, if successful, provisions the Kafka topic and ACL in PP.
3. A pull request to `main` triggers further validation; upon approval, resources are created in PRD, and documentation is updated automatically.
4. The event flow becomes discoverable and ready for use, and the team requests any required access credentials through standardized, auditable channels.

This process illustrates the empowerment of LoB teams to manage their own integrations, reducing reliance on central IT and improving responsiveness.

4.2 MFT Automation

The MFT platform represents the first realization of the vision for modern Integration Platforms at Sonae MC, enabling a standardized, declarative, and autonomous approach to file transfer processes. The core of the solution is based on Fortra's GoAnywhere MFT and GoAnywhere Gateway products, replacing the legacy SOA-based file transfer solution. The architecture (Figure 4.7) supports both pre-production (PP) and production (PRD) environments, with high availability clusters initially located within the same datacenter for support reasons.

All communication from the MFT cluster nodes, regardless of destination, is routed through the Gateway, ensuring consistent governance and security. The platform supports hybrid deployment models (on-premises and cloud) and integrates with modern storage solutions such as Azure Blob Storage and Amazon S3, in addition to traditional protocols like FTP and SFTP. File transfers can be triggered by polling, external REST API invocations, or scheduled workflows. Monitoring is implemented through Grafana, while execution logs are available through the GoAnywhere UI, providing transparency and operational insight.

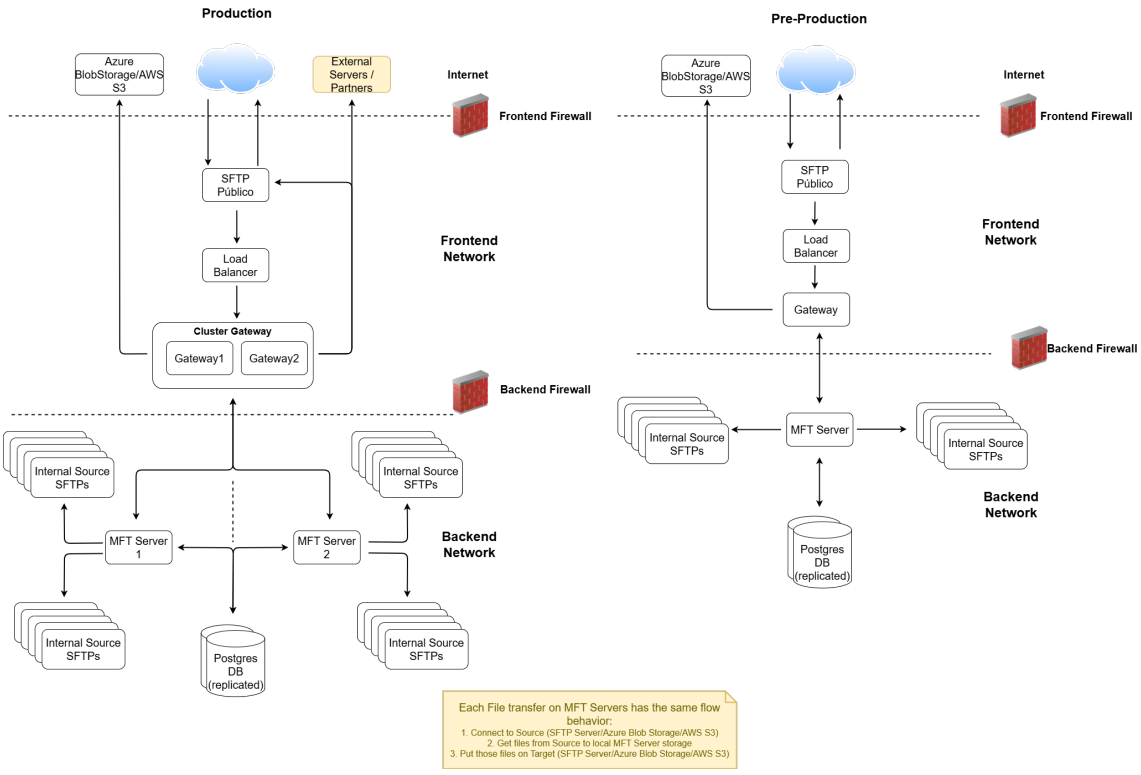


Figure 4.7: MFT Architecture.

A key design principle was to first replicate the capabilities of the legacy "Connectivity" framework within GoAnywhere, while significantly improving developer experience with self-service and automation. As a result, the MFT Platform currently supports three primary file transfer scenarios:

- **One to One**: Automates the transfer of files from a single source to a single destination. The workflow involves connecting to the source resource, collecting files according to specified parameters, transferring them to the MFT platform, connecting to the target resource, and finally transferring the collected files to the target. Deployment of this scenario requires one project specification file, one source resource specification, and one target resource specification, each referenced in the main project YAML.

- **One to Many**: Supports distributing files from a single source to multiple destinations in a single automated workflow. The process starts by connecting to the source, collecting the required files, transferring them to the MFT platform, and then iterating through each target resource to transfer the files accordingly. To deploy, this scenario requires one project specification file, one source resource specification, and as many target resource specification files as there are destinations. Each is referenced in the project's YAML configuration.

- **Store Templates**: This scenario streamlines the onboarding of similar file transfer projects by leveraging reusable templates, particularly for large-scale use cases such as multi-store integrations. By abstracting common configurations, store templates enable rapid, consistent, and scalable setup of multiple projects with minimal manual intervention.

Configuration of new file transfer processes (Figure 4.8) is fully declarative, relying on YAML templates versioned in a central GitHub repository. Each YAML specification describes the source and destination endpoints, required secrets (referenced via HashiCorp Vault), scheduling/triggers (such as cron or polling), file naming and post-processing rules, and owner/incident support teams. Advanced use cases such as multi-protocol delivery, cloud storage integration, and custom error handling are supported through modular YAML templates. The structure of the repository and the usage of separate rules repositories ensures maintainability, quality control, and knowledge sharing.

Before onboarding a new specification, teams must ensure that all required Jira tickets are created, the necessary secrets are securely stored in HashiCorp Vault, appropriate firewall rules are open, and any new endpoints are registered in the Configuration Management Database (CMDB). The CMDB is a centralized repository that maintains detailed information about the organization's IT assets, including hardware, software, network components, and their relationships and dependencies. Registering endpoints in the CMDB ensures that all infrastructure components involved in the file transfer are properly catalogued, tracked, and governed, thereby enhancing traceability, facilitating impact analysis, and supporting compliance with enterprise IT policies throughout the lifecycle of every file transfer integration.

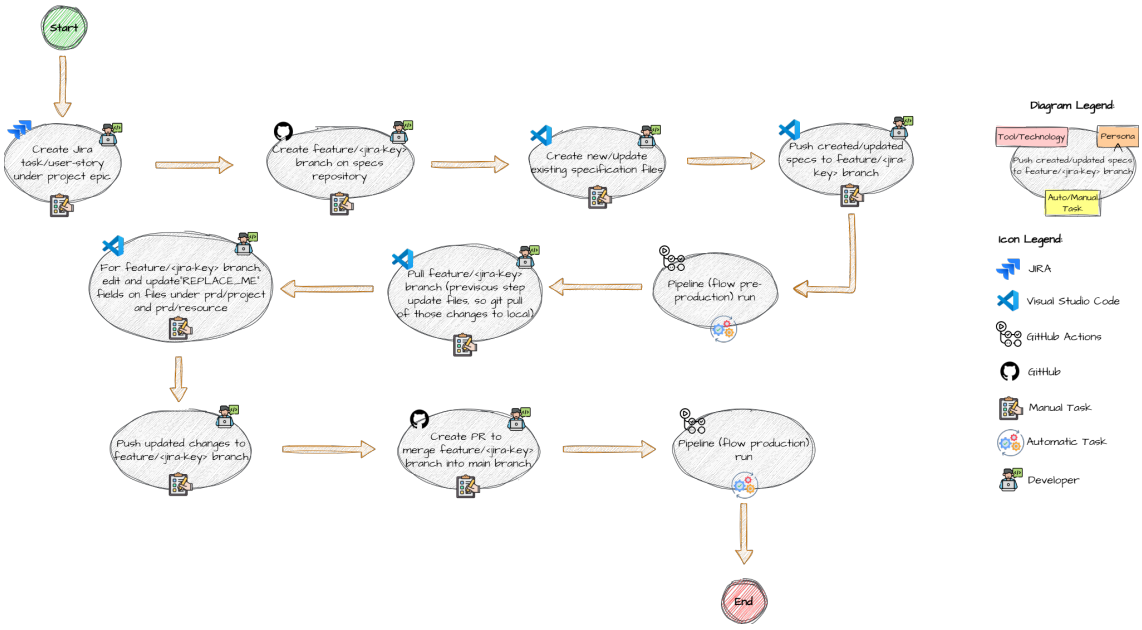


Figure 4.8: MFT Macro Workflow.

Pipeline

The automation pipeline for MFT is designed to enforce organizational standards, streamline development, and provide operational reliability through comprehensive validation, automation, and governance mechanisms. The entire deployment lifecycle is grounded in GitOps principles, where all changes to transfer specifications are managed as code, versioned, and promoted through well-defined pipelines.

Every time a new branch is created in GitHub, each commit triggers the pre-production (PP) pipeline (Figure 4.9). This pipeline performs extensive validation of all YAML files using Spectral, ensuring compliance with organizational naming conventions, completeness of mandatory fields, and the absence of orphaned resources—that is, resources not linked to a valid project. The pipeline also analyzes the impact of the proposed development: for new resources, it confirms that there are no side effects, while for modifications, it identifies and communicates all affected dependencies and resources to the developer.

Connectivity tests are conducted to ensure network permissions are sufficient for the intended transfers. If a firewall rule is missing, the pipeline automatically sends an email to the responsible team with a pre-filled template to request the necessary firewall rule within the organization. Only after successful validation does the workflow proceed to create the required objects in the MFT platform.

To maintain repository consistency, the pipeline automatically renames all relevant files according to the established conventions using a dedicated GitHub bot. This bot also initializes the corresponding production (PRD) files based on the validated PP configuration, allowing developers to focus solely on completing any PRD-specific details. Once ready, developers open a

pull request from their feature branch to `main`, which triggers the PRD pipeline (Figure 4.10). The PRD pipeline mirrors all validation and deployment steps of PP but only provisions PRD resources after the pull request is merged.

Reviewer assignment is fully automated: the pipeline extracts the responsible team's email from the specification, matches it to the correct GitHub team, and assigns a reviewer from each affected team. If an assigned reviewer is unavailable, any member of the respective team can approve the pull request, ensuring development is not blocked by individual absence. All validation feedback and impact analysis are provided directly in the pull request for rapid iteration and issue resolution.

Upon successful deployment in either PP or PRD, the pipeline sends an environment-specific CAWA template by email. In the event of errors—such as firewall issues—the pipeline immediately notifies the relevant team and provides actionable templates to expedite resolution. In future iterations, the goal is to integrate a native CAWA agent within the MFT platform to eliminate the need for manual template distribution, further streamlining operational workflows.

Secret management is handled securely via HashiCorp Vault, with each team having access only to its own secrets. This strict access control model ensures that credentials and sensitive data are never exposed in code or logs, supporting both security and compliance. Secrets are referenced by path in the YAML specs and injected dynamically at runtime, facilitating secure rotation and least-privilege operations.

Deployment and orchestration are achieved via REST API calls to GoAnywhere, with all infrastructure dependencies—such as firewall rules, hosts, and storage endpoints—explicitly referenced in the configuration. Scheduling can be defined declaratively through cron expressions or polling intervals, supporting both periodic and event-driven file transfers. The use of Docker for GoAnywhere components also eases upgrades and reduces technical debt.

Operational monitoring is provided through Grafana dashboards, giving teams real-time visibility into platform health and transfer job status. CAWA ensures timely automated alerts and incident management for both PP and PRD environments. Detailed execution logs are available in the GoAnywhere UI for rapid troubleshooting and post-mortem analysis.

Finally, the platform is architected for resilience, with comprehensive error handling and recovery mechanisms. All execution errors are logged and trigger notifications. Teams are empowered to access logs and independently perform recovery actions, such as retrying or resuming transfers. File renaming conventions—such as marking files as `.lock` during processing and `.ok` upon transfer completion—and directory archiving facilitate atomic operations and support safe, automated recovery even in cases of interrupted or partial transfers.

This tightly integrated pipeline ensures that only high-quality, compliant, and fully reviewed configurations are deployed to the MFT platform, enabling secure, auditable, and autonomous management of file transfers across the enterprise.

Real-World Example

A typical onboarding scenario involves a team needing to automate the delivery of sales reports from an internal application to an external partner's SFTP server and to an Azure Blob Storage bucket. The team creates a YAML specification defining the source, both destinations, required credentials (referenced in Vault), scheduling, and post-processing rules. They submit a pull request in the lac1011-mft-specs repository. The pipeline validates the spec, ensures all dependencies are met, and, upon approval, deploys the project to GoAnywhere. From Grafana, the team can monitor each transfer, while CAWA notifies them of any issues. If a failure occurs, logs are available in GoAnywhere, and the team can independently retry or recover the failed files without intervention from the Integration Platforms team.

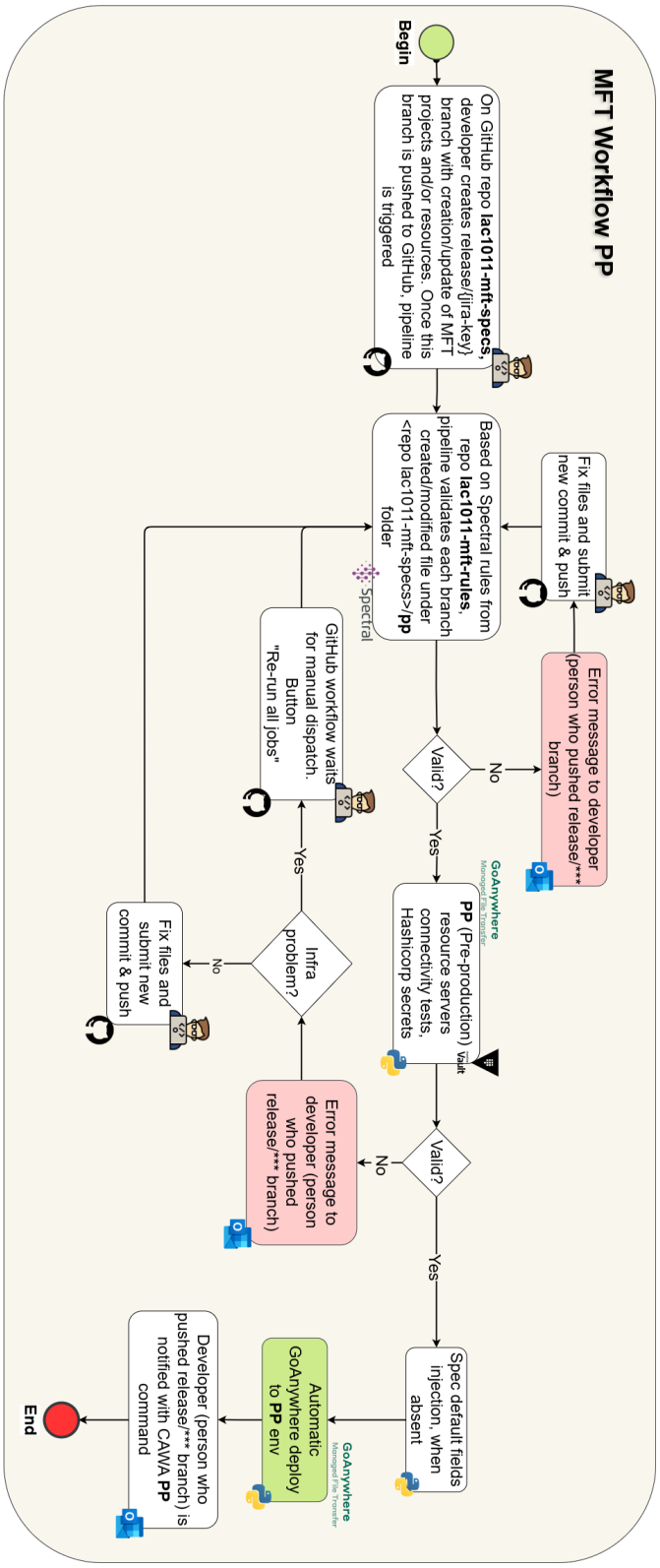
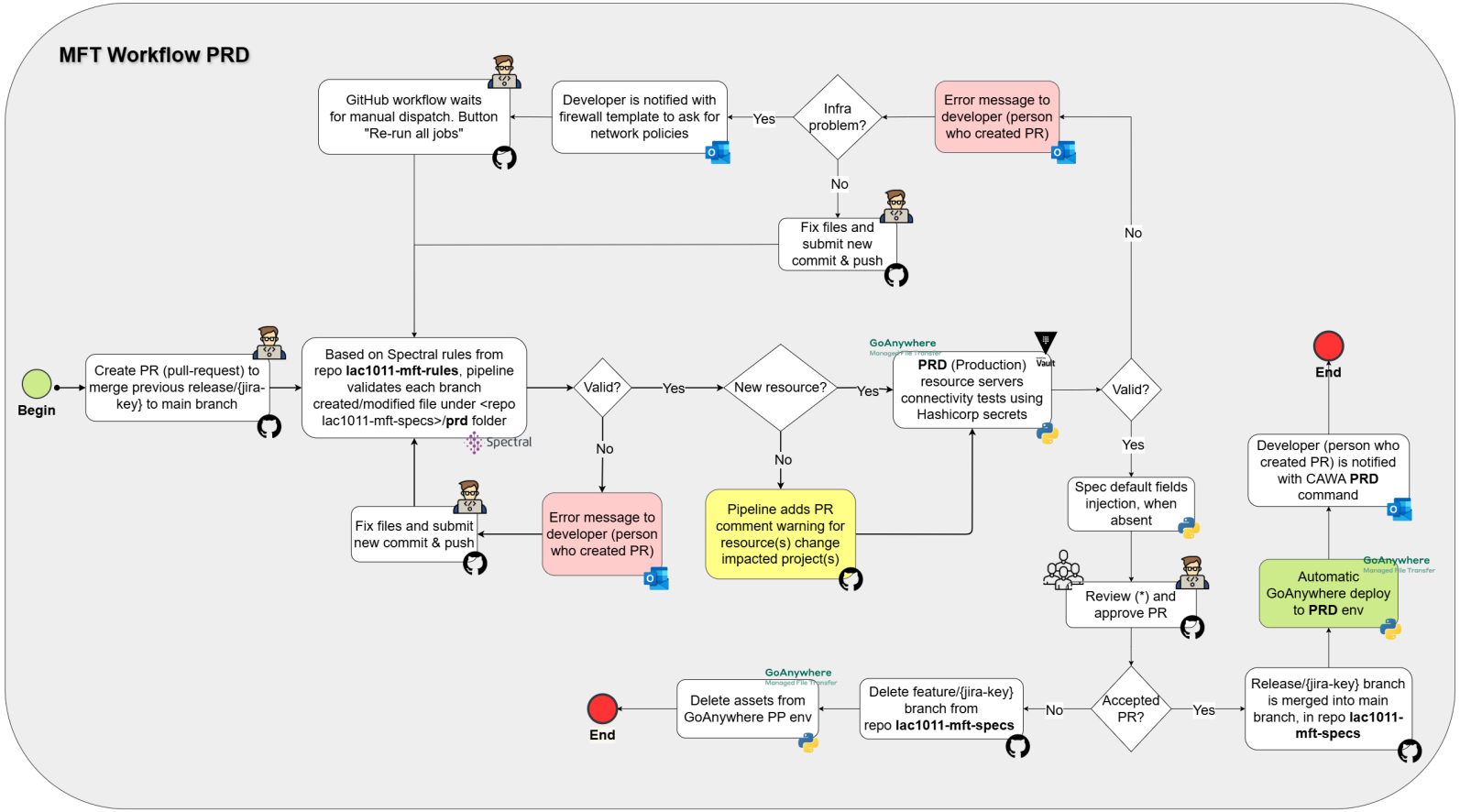


Figure 4.9: MFT Pipeline Workflow PP.



(*)

1st phase (Pilot): 1 Integration Platforms team reviewer + n reviewers (1 per each vertical stream impacted by development - streams identified by provided spec LACIds)

2nd phase: n reviewers (1 per each vertical stream impacted by development - streams identified by provided spec LACIds)

Legend:

- Outlook
- Github
- Spectral
- Python
- Developer Interaction
- Manual Interaction of reviewers
- Hashicorp Vault
- GoAnywhere Managed File Transfer

Figure 4.10: MFT Pipeline Workflow PRD.

Chapter 5

Results and Discussion

This chapter presents a comparative analysis of the performance improvements and operational gains achieved through the implementation of two key automation artefacts at Sonae MC: the migration from the legacy SOA-based "Connectivity" solution to the new MFT platform, and the deployment of a self-service EDA automation framework. The evaluation encompasses both quantitative and qualitative dimensions, providing a holistic perspective on how these artefacts have transformed integration practices, accelerated delivery cycles, and strengthened governance across the organization.

Before presenting specific outcomes and user experiences, it is important to clarify that a qualitative analysis was conducted to complement and reinforce the quantitative findings. This approach was particularly essential in the context of the EDA artefact, where it was not feasible to extract meaningful quantitative metrics, making qualitative feedback from real users and stakeholders indispensable for evaluating the impact of the solution. Throughout the evaluation phase of this project, I conducted interviews with real stakeholders who directly interacted with the new integration solutions at MC Digital. To preserve confidentiality and respect internal policies, the names of the interviewees are not disclosed instead, each individual is represented by a pseudonym. These personas are grounded in authentic feedback and real user stories, serving to humanize the impact that the EDA and MFT artefacts have had on a diverse range of teams and roles within the company.

Together, the results for both artefacts validate the strategy of democratizing integration capabilities through automation, self-service, and robust governance mechanisms. The findings highlight not only dramatic gains in raw performance and operational throughput, but also deeper cultural and structural benefits—such as increased autonomy for business teams, reduced reliance on central IT, and the fostering of a continuous improvement mindset across the integration landscape at Sonae MC.

5.1 EDA: User Experience and Developer Agility

While file transfer modernization brought visible operational gains, the introduction of the EDA self-service automation framework fundamentally changed the developer experience and the organization's agility in building event-based integrations. To better illustrate these improvements, this section presents three personas—each representing a typical developer or team member who interacted with the EDA automation solution.

Persona 1: Ana, the Application Developer (LoB)

Ana is an application developer responsible for integrating her team's retail analytics microservice with enterprise-wide data flows. Before the EDA automation, Ana had to submit requests to the Integration Platforms team to provision Kafka topics, manage ACLs, and register events. This process could take several days or even weeks, often requiring multiple clarification emails, manual validations, and long waiting times for approvals.

With the adoption of the EDA self-service pipeline, Ana now uses YAML templates to define event specifications, including metadata, schema, and ownership. She submits these through a GitHub pull request, where automated validation scripts instantly check compliance with organizational rules and schemas. Errors and suggestions are provided directly in the PR, allowing Ana to iterate and correct issues within minutes. Once approved, the pipeline automatically provisions the Kafka resources and updates the Event Portal documentation. Ana receives notifications upon completion and can start consuming or producing events in hours rather than days, dramatically accelerating her team's development lifecycle.

Persona 2: Miguel, the Integration Specialist

Miguel is a senior integration specialist who frequently assists various business units in onboarding new event streams or troubleshooting issues with data flows. In the pre-EDA scenario, Miguel was often inundated with repetitive support tickets: manual reviews of YAML/JSON event specs, checking for naming convention compliance, and updating documentation across disparate systems.

With the new automation, Miguel's focus has shifted from repetitive, low-value manual tasks to higher-level governance and best-practice evangelization. The automated pipeline relieves him from hand-checking specs; instead, he can spend his time improving validation rules, supporting onboarding workshops, and analyzing audit logs for continuous improvement. Moreover, Event Portal's live documentation ensures that Miguel, as well as other stakeholders, have a real-time, single source of truth for all event and service integrations.

Persona 3: Joana, the Data Engineer

Joana is a data engineer responsible for onboarding new producers and consumers to Kafka for a marketing analytics use case. Previously, coordinating with central teams to set up topics and

ACLs often created delays in her project timelines, especially when errors were discovered late in the process. The lack of real-time feedback meant that simple mistakes (e.g., a missing mandatory field or schema mismatch) could result in several rounds of back-and-forth communication.

With the EDA self-service model, Joana can independently create and test new event flows by leveraging the validation pipeline. The feedback is immediate and actionable, allowing her to fix issues early. Documentation and access are handled automatically, and Joana can verify through Event Portal that all producer/consumer relationships are properly registered and discoverable. She notes a significant reduction in project lead time and an increase in her autonomy and confidence when delivering new data pipelines.

Discussion: EDA Platform Outcomes

The introduction of the EDA automation solution has transformed not only technical processes but also the organizational culture and day-to-day experience of developers, integration specialists, and data engineers at Sonae MC. The following discussion, grounded in the practical realities of Ana, Miguel, and Joana, highlights how the new self-service platform has fundamentally improved operational efficiency, collaboration, and job satisfaction across multiple teams.

Dramatic Reduction in Onboarding Time: Prior to automation, the end-to-end process of onboarding a new event—encompassing everything from initial requirements gathering to topic creation, ACL configuration, and documentation—was often characterized by protracted delays and repetitive handoffs. As Ana experienced, developers would wait several days or even weeks for each new integration, sometimes blocked by simple misunderstandings or missing information. Analysis of Jira metrics confirmed this reality: under the previous CoE model, the time to market for delivering a new integration flow—including development, testing, and production rollout—ranged from 2 to 5 weeks. With the introduction of the C4E model and the self-service automation framework, this timeline was reduced dramatically to just 2 to 3 days end-to-end. The new framework has collapsed these timelines; what once required extensive coordination with central IT now takes place in a matter of hours, as validation, provisioning, and documentation are triggered automatically from a single pull request. Developers like Ana can move quickly from ideation to production, dramatically accelerating time-to-market for new features and products.

Self-Service Empowerment and Increased Autonomy: The most profound change for end users is the shift to true self-service. Previously, Joana's work as a data engineer was often interrupted by external dependencies—waiting for approval, chasing support for basic changes, or lacking visibility into current integration status. With the EDA pipeline, she is now empowered to iterate independently: she can design new events, instantly validate them against organizational standards, and deploy them into pre-production and production environments without leaving the GitHub interface. This has given teams like hers the autonomy to innovate at their own pace, while reducing the "friction cost" of collaboration with central integration teams.

Governance and Compliance at Scale: For integration specialists such as Miguel, the automation framework has shifted the focus from "gatekeeping" to "enabling." Instead of policing

compliance manually—checking YAML files, reviewing naming conventions, or updating documentation—Miguel now trusts the automated validation logic embedded in the pipeline. This ensures every artefact, regardless of the team or technical maturity of its author, meets the required standards before it ever reaches production. As a result, the quality and consistency of integrations have measurably improved, even as the volume of requests has increased. For management and IT auditors, this represents a significant gain in risk mitigation, as every change is systematically logged, reviewed, and documented.

Transparency, Discoverability, and Cross-Team Collaboration: A central pain point in the previous model was the fragmentation of documentation and poor visibility into existing integrations. With real-time updates to the Event Portal, Ana, Miguel, Joana, and all other stakeholders can now easily search, browse, and audit the full set of active event flows, producers, and consumers across the organization. This has led to a more collaborative environment: teams are less likely to "reinvent the wheel," as they can quickly discover reusable events, schemas, and services. Onboarding new team members is also easier, as all knowledge is centrally documented and discoverable, rather than hidden in emails or siloed spreadsheets.

Faster Feedback, Better Quality, and Iterative Improvement: A key feature lauded by all three personas is the immediacy of feedback. If a YAML specification is incomplete, a schema is invalid, or a required field is missing, the pipeline provides actionable feedback directly in the pull request. This "shift-left" approach to quality control ensures that errors are caught early and iteratively resolved by the author, rather than being discovered days later by a separate team. Developers feel ownership of their work, and the organization benefits from higher-quality, more robust integrations.

Cultural Transformation and Motivation: Perhaps the most important, though least quantifiable, impact of the EDA automation framework is the cultural transformation it has triggered. The experience of Miguel—who now dedicates time to training, template improvement, and best practice dissemination—illustrates a broader trend: teams are shifting from reactive and manual support to proactive enablement and shared responsibility for integration excellence. There is a new sense of pride and ownership among LoB developers, who see the platform as an enabler of creativity rather than a bureaucratic hurdle. Continuous improvement has become part of the workflow, as feedback from real usage is quickly incorporated into scripts, validation rules, and onboarding guides.

Organizational Impact and Scalability: The compounding effect of these changes is clear: as more teams adopt the EDA self-service model, the organization becomes more agile, resilient, and scalable. Miguel and his colleagues in the integration platforms team can focus on platform evolution, security, and governance, rather than being overwhelmed by support tickets. Developers and data engineers deliver more value, faster, and the organization is better equipped to respond to new business opportunities or challenges.

A Model for Continuous Democratization: The experiences of Ana, Miguel, and Joana demonstrate that well-designed automation and self-service not only solve technical problems, but can also democratize access to integration capabilities, foster collaboration, and drive a pos-

itive cultural shift (Forsgren et al., 2018). The success of the EDA automation artefact provides a replicable blueprint for other domains—such as MFT, APIs, or even infrastructure management—further extending the benefits of decentralization and automation across the enterprise.

In summary, the EDA platform at Sonae MC is not simply a technical upgrade; it is a strategic enabler for organizational agility, developer satisfaction, and future-proof integration governance.

5.2 MFT: Quantitative and Qualitative Results

For the MFT evaluation, three representative integration flows previously handled by the Connectivity framework were re-implemented and executed on the GoAnywhere MFT platform. The scenarios cover both one-to-one and one-to-many transfers, representing typical batch file movements between internal systems, external partners, and cloud storage.

Transfer times were measured for both platforms in pre-production, using identical network conditions and endpoints. Table 5.1 summarizes the observed transfer times and file characteristics for each evaluated scenario.

Table 5.1: Performance comparison between Connectivity and MFT platforms

Scenario	Platform	Files	Total Size (bytes)	Time (seconds)
FO.AZURESANE.CONTROL_ROLLERS	Connectivity	1	40,547	20
	MFT	1	596,142	4
INSAPFI.SAPFI.AccountingData	Connectivity	287	434,600	105
	MFT	317	1,359,344	16
FO.AZURESANE.BOX_SHEETS	Connectivity	1	77,696	35
	MFT	1	157,076	2

As the results indicate, the MFT platform dramatically outperforms the legacy solution in every scenario, delivering both higher throughput and lower latency. Even when handling larger volumes of data, transfer times were reduced by factors ranging from 5x to more than 10x, reflecting the impact of modernized engine architecture, parallelism, and optimized protocols.

- For the **FO.AZURESANE.CONTROL_ROLLERS** scenario, transferring a much larger file (596,142 bytes) via MFT required only 4 seconds, compared to 20 seconds for a much smaller file (40,547 bytes) in Connectivity.
- The **INSAPFI.SAPFI.AccountingData** flow, involving hundreds of files, saw the total transfer time reduced from 1 minute and 45 seconds (105 seconds) for 287 files (434,600 bytes) in Connectivity to only 16 seconds for 317 files (1,359,344 bytes) in MFT, demonstrating vastly superior throughput and parallelization capabilities.
- In the **FO.AZURESANE.BOX_SHEETS** test, the MFT solution delivered a file of 157,076 bytes in just 2 seconds, whereas Connectivity took 35 seconds for a file less than half that size.

These gains reflect the architectural advantages of the new platform, including parallelism, protocol optimization, and automation.To complement the quantitative analysis, the following personas illustrate the qualitative benefits and organizational impact of the new MFT solution, each representing a distinct stakeholder group within Sonae MC.

Persona 1: Carlos, Business Process Owner

Carlos oversees the delivery of financial and operational reports to external partners and central data repositories. Under the previous system, he depended on the IT department for even minor changes to file transfer flows—a process fraught with delays, manual handovers, and unclear requirements. With the new MFT platform, Carlos can now define new transfer specifications using guided YAML templates, submit requests via Git, and receive immediate, actionable feedback from automated validations. Once approved, new file transfer flows are provisioned and operational within hours. This enables Carlos to respond rapidly to business demands, introduce new integrations, and reduce the time-to-market for critical initiatives.

Persona 2: Beatriz, IT Integration Engineer

As an integration engineer, Beatriz was once immersed in a constant cycle of debugging scripts, resolving configuration errors, and managing repetitive onboarding requests from the business. The transition to MFT automation has fundamentally changed her role. Beatriz now focuses on enhancing validation rules, developing onboarding documentation, and mentoring business users on best practices. The substantial reduction in manual tasks and troubleshooting has allowed her to concentrate on more strategic, value-adding activities. She reports a notable decrease in support tickets and an improvement in job satisfaction, as the integration team now serves as an enabler and advisor rather than a bottleneck.

Persona 3: Ricardo, Tech Lead

Ricardo leads the operations team responsible for the continuous, secure flow of data across dozens of internal and external endpoints. Previously, operational visibility was limited, and error handling often involved time-consuming coordination with IT. With GoAnywhere MFT, Ricardo and his team now have real-time access to dashboards in Grafana, benefit from proactive incident alerts via CAWA, and can independently monitor, troubleshoot, and resolve issues. The versioned configuration in Git ensures traceability, while standardized templates accelerate onboarding of new team members. Ricardo credits the new solution with reducing operational stress, increasing his team's autonomy, and elevating overall service reliability.

Discussion: MFT Platform Outcomes

The performance gains observed with the MFT platform are attributable to several architectural and operational improvements:

- **Modernized engine**: GoAnywhere MFT is optimized for high-throughput, parallel file operations, and handles larger payloads efficiently without incurring the latency typical of legacy SOA frameworks.

- **Declarative configuration and automation**: Self-service onboarding, automated validation, and error handling allow for streamlined development and fewer manual interventions.
- **Network and protocol optimization**: The platform supports advanced connection pooling and more efficient session management, reducing the overhead of connection setup and teardown for each file.
- **Improved error recovery and monitoring**: Real-time dashboards and alerting reduce downtime, while better logging accelerates root cause analysis in case of failure.

Besides raw speed, the new MFT approach offers qualitative advantages:

- **Agility**: Teams can onboard, test, and deploy new integrations much faster, reducing time-to-market for new projects.
- **Security and governance**: Automated secret management with HashiCorp Vault and robust audit trails support compliance and minimize operational risks.
- **Scalability**: The solution seamlessly scales to handle more concurrent transfers, larger payloads, and more complex delivery patterns (one-to-many, multi-cloud, etc.).
- **Self-service and transparency**: The declarative, GitOps-based process enables business and technical users to manage their own file flows, with less dependency on central teams.

The migration from SOA to MFT introduced a diverse set of challenges, spanning technical, cultural, and operational domains. One of the initial hurdles was the need to educate both technical and non-technical teams in modeling file transfer specifications using YAML. This required extensive documentation, hands-on workshops, and the creation of guided templates to lower the entry barrier and promote best practices in declarative modeling.

Managing secrets and network permissions also proved complex at first. Teams had to adapt to using HashiCorp Vault for credential management, which, although significantly improving security, demanded a new understanding of access policies, secret rotation, and least-privilege principles. Similarly, the introduction of automated network tests and firewall rule requests in the pipeline helped to catch misconfigurations early, but necessitated tight coordination between development, security, and infrastructure teams. Ensuring that all required firewall rules were in place prior to onboarding became an operational priority, and the pipeline's ability to generate actionable email templates for missing rules streamlined this process.

Another challenge was the shift from ad hoc, manual monitoring and error handling to the use of centralized dashboards (Grafana) and incident automation (CAWA). While these tools significantly improved observability and response times, they required teams to develop new skills in interpreting dashboards and leveraging automated alerts. The need for robust error recovery led to the formalization of file renaming conventions and automated archival strategies, ensuring that failures could be handled without manual intervention or data loss.

Legacy integrations posed additional obstacles. In some cases, existing file transfer logic or partner requirements did not align with the new MFT paradigm, necessitating the development of custom templates, transitional automation scripts, or phased migration approaches. These scenarios highlighted the importance of flexibility in the onboarding process and the need for clear escalation paths when edge cases were encountered.

Despite these challenges, the transition has delivered substantial benefits. Teams now onboard and manage file transfers with far greater agility and autonomy, reducing their reliance on central IT and support teams. Notably, the time to market for implementing new file transfer solutions—which previously ranged from 3 to 6 weeks under the centralized CoE model—has been reduced to just 2 to 5 days with the adoption of the C4E self-service approach. The self-service, GitOps-driven model has made processes more transparent and auditable, while automated validations and reviews have improved quality and compliance. Security posture is strengthened by granular access controls, and operational incidents are identified and addressed more rapidly.

Crucially, a culture of continuous improvement has taken root. Ongoing feedback from users has resulted in regular enhancements to YAML templates, validation rules, automation scripts, and onboarding documentation. This iterative approach ensures that the platform continues to evolve in response to real-world usage, positioning the MFT solution as a resilient and scalable foundation for future integration needs across the organization.

In summary, the migration to the GoAnywhere MFT platform has resulted in:

- Drastic reductions in file transfer times (up to 10x or greater in tested scenarios)
- Enhanced reliability and transparency of operations
- Higher throughput, even as file sizes and file counts increased
- More efficient onboarding and maintenance of file transfer processes

These results validate the strategy of replacing the legacy SOA/Connectivity solution with a modern, declarative, and automated MFT platform. By leveraging these gains, Sonae MC can better support evolving business requirements and further democratize technical development in its integration landscape.

Chapter 6

Conclusion

The work presented in this dissertation set out to address a central challenge faced by large, digitally mature organizations: how to accelerate, democratize, and govern enterprise integration in environments marked by heterogeneity and scale. To this end, two artefacts were designed, implemented, and evaluated within MC Digital at Sonae MC, each representing a concrete, rigorously validated solution to a class of integration problems.

6.1 Contributions

The first artefact, an automated Event-Driven Architecture (EDA) framework, introduced a self-service paradigm for event integration by empowering Lines of Business (LoB) to independently define, validate, and deploy event flows using standardized YAML specifications, Python automation, and robust CI/CD pipelines. This artefact transformed the developer experience, as documented through quantitative results and persona-driven qualitative analysis: onboarding times for new event streams were reduced from weeks to hours, and the shift from centralized gatekeeping to decentralized enablement fostered both agility and a culture of proactive improvement. Centralized governance was preserved through automated validation, integrated documentation, and systematic audit trails, ensuring that the democratization of integration did not compromise compliance or security. As demonstrated, this artefact provides a scalable model for enabling innovation and reducing bottlenecks, while embedding best practices and organizational standards into every stage of the integration lifecycle.

The second artefact, a modernized MFT automation platform, replaced a legacy SOA-based solution with a declarative, GitOps-driven pipeline for secure, auditable, and self-service file transfer. Through the use of YAML-based configuration, automated validation with Spectral, secrets management with HashiCorp Vault, and integrated monitoring with Grafana and CAWA, this artefact empowered both technical and business stakeholders to design, deploy, and operate complex file flows rapidly and securely. Quantitative experiments evidenced dramatic improvements in throughput and latency, while qualitative analysis—supported by personas representing business process owners, integration engineers, and operations leads—demonstrated significant reductions

in time-to-market, greater transparency, and enhanced job satisfaction. As with the EDA artefact, governance, security, and traceability remained foundational, with all changes version-controlled and subject to systematic review.

Taken together, these artefacts exemplify the practical realization of C4E principles within a large-scale enterprise. The research demonstrates that it is possible to decentralize integration responsibilities, enable business-driven innovation, and maintain rigorous oversight—provided that automation, validation, and self-service are embedded as architectural pillars. The impact of this work is seen not only in performance metrics, but also in cultural transformation: business and technical teams alike reported increased autonomy, reduced friction, and a greater sense of ownership over integration outcomes.

Several important lessons emerge from this dual-artefact approach:

- **Declarative, Versioned Specification**: Both frameworks rely on human-readable, version-controlled artefacts as the primary interface, ensuring transparency, traceability, and ease of collaboration.
- **Automated Validation and Governance**: The enforcement of standards, naming conventions, and compliance policies through code, rather than manual review, has proven essential for scalability and risk mitigation.
- **Separation of Concerns and Modularity**: Decoupling specification, validation, provisioning, and monitoring simplifies onboarding, supports iterative improvement, and enables rapid adaptation to changing requirements.
- **Continuous Feedback and User-Centered Design**: The iterative refinement of templates, pipelines, and onboarding materials, based on direct user feedback, was critical to adoption and success.
- **Cultural and Organizational Impact**: Automation is not solely a technical innovation—it is a catalyst for broader change in how teams collaborate, innovate, and assume responsibility for integration excellence.

6.2 Future Work

While the results achieved are significant, there are clear avenues for further research and development:

- **Unified Integration Dashboards**: Future work could focus on the development of integrated dashboards that unify event-driven and file-based flows, providing holistic visibility, analytics, and incident response across all integration artefacts.
- **Expanding Protocol and Platform Support**: Both frameworks can be extended to support additional protocols, cloud platforms, and integration patterns, further broadening their applicability.

- **Automated Dependency Management**: Enhancements to impact analysis and dependency detection could further streamline change management and reduce operational risk, especially in large-scale environments with many interconnected artefacts.
- **Intelligent Recommendation and Error Correction**: Leveraging machine learning to provide automated suggestions, error remediation, and best practice recommendations during artefact onboarding could further lower the barrier for non-specialist users.
- **Organizational Training and Change Management**: Continued investment in training, onboarding, and community building is necessary to sustain adoption and maximize the benefits of democratized integration.

By combining self-service, automation, and rigorous governance, the solutions implemented at MC Digital illustrate a path toward increased agility, reduced operational friction, and scalable, future-proof digital transformation. As digital ecosystems continue to evolve, the principles and practices demonstrated here will remain essential for organizations seeking to balance innovation with control, and to empower teams at every level to contribute to integration excellence.

References

AboElHassan, A., & Yacout, S. (2023). A digital shadow framework using distributed system concepts. *Journal of Intelligent Manufacturing*, *34*, 3579–3598.

Bhat, M., & H, V. P. (2024). Access point management using java based microservice. *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)*, 1–5.

Breunig, D. A., Roedel, A., & Bauernhansl, T. (2020). Extending service-oriented architectures in manufacturing towards fog and edge levels. *IEEE International Conference on Industrial Informatics (INDIN)*, *2020-July*, 291–298.

Bruns, R., & Dunkel, J. (2014). Towards pattern-based architectures for event processing systems. *Software - Practice and Experience*, *44*, 1395–1416.

Chukhajyan, H. (2016). Automated generation of core test description file for hierarchical test. *Proceedings of 2016 IEEE East-West Design and Test Symposium, EWDTS 2016*.

Dhamdhere, D. M. (2020). A systematic approach to integration challenges in modern enterprises. *International Journal of Computer Applications*.

Estruch, A., & Álvaro, J. A. H. (2012). *Event-driven manufacturing process management approach* (Vol. 7481 LNCS).

Ferrer, B. R., Iarovyi, S., Lobov, A., & Lastra, J. L. M. (2015). Towards processing and reasoning streams of events in knowledge-driven manufacturing execution systems. *Proceeding - 2015 IEEE International Conference on Industrial Informatics, INDIN 2015*, 1075–1080.

Fishman, S. (2022). Coe vs. c4e vs. away teams: Aligning speed, scale, and safety [Available at https://blogs.mulesoft.com/digital-transformation/coe-c4e-away-teams/, last accessed in Jun 2025].

Folds, D. J., & McDermott, T. A. (2019). The digital (mission) twin: An integrating concept for future adaptive cyber-physical-human systems. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 748–754.

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The science of lean software and devops*. IT Revolution Press.

Hevner, A. (2007). A three cycle view of design science research. *Scandinavian Journal of Information Systems*, *19*.

Hyvönen, P., Vidmark, A., Andersson, M., Liljeblad, M., Alvarez, J. M., & Massey, D. (2018). Development of the infinity service, a data-centric ground network service with high capacity for small satellites and large constellations. *15th International Conference on Space Operations, 2018*.

Khriji, S., Benbelgacem, Y., Chéour, R., Houssaini, D. E., & Kanoun, O. (2022). Design and implementation of a cloud-based event-driven architecture for real-time data processing in wireless sensor networks. *Journal of Supercomputing*, *78*, 3374–3401.

Krishnan, M. G., Vijayan, A. T., & Ashok, S. (2022). *Real-time integration of industrial robot with matlab* (Vol. 828).

Laoyan, S. (2024). What is agile methodology? (a beginner's guide) [Available at https://asana.com/pt/resources/agile-methodology, last accessed in Jun 2025].

Lopes, P., & Oliveira, J. L. (2015). *An event-driven architecture for biomedical data integration and interoperability* (Vol. 9044).

Molina, J. M., García, J. F., & Jiménez, C. K. (2018). Archer: An event-driven architecture for cyber-physical systems. *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, 335–340.

Narkhede, N., et al. (2011). Kafka: A distributed messaging system for log processing. *ACM*.

Ovington, T. (2023). What is a line of business (lob) + examples? [Available at https://www.digital-adoption.com/line-of-business/, last accessed in Jun 2025].

Pan, H., Chard, R., Zhou, S., Kamatar, A., Vescovi, R., Hayot-Sasson, V., Bauer, A., Gonthier, M., Chard, K., & Foster, I. (2024). Octopus: Experiences with a hybrid event-driven architecture for distributed scientific computing. *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 496–507.

Papazoglou, M. P., & van den Heuvel, W.-J. (2007). Service oriented architectures: Approaches, technologies and research issues. *The VLDB Journal*, *16*(3), 389–415.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77.

Peng, L., Tang, X., Li, C., Xiao, Y., Zhang, T., & Weng, Y. (2023). Decentralized reputation-based leader election for privacy-preserving federated learning on internet of things. *2023 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, 362–369.

Romero, E. E., Camacho, C. D., Montenegro, C. E., Acosta, Ó. E., Crespo, R. G., Gaona, E. E., & Martinez, M. H. (2022). Integration of devops practices on a noise monitor system with circleci and terraform. *ACM Trans. Manage. Inf. Syst.*, *13*.

Ross, J. W., Weill, P., & Robertson, D. C. (2006). *Enterprise architecture as strategy: Creating a foundation for business execution*. Harvard Business Review Press.

Seetharamugn. (2023). What is an event-driven architecture? [Available at https://medium.com/@seetharamugn/the-complete-guide-to-event-driven-architecture-b25226594227, last accessed in Jun 2025].

Vangala, S. R., Kasimani, B., & Mallidi, R. K. (2022). Microservices event driven and streaming architectural approach for payments and trade settlement services. *2022 2nd International Conference on Intelligent Technologies (CONIT)*, 1–6.

Appendix A

Literature Review: PRISMA Methodology

To ground the research in current scientific and industry knowledge, a comprehensive literature review was conducted using the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) methodology. PRISMA offers a transparent and reproducible process for identifying, screening, and selecting relevant studies, ensuring that this work builds upon validated findings and avoids duplication of effort.

A.1 Research Scope

This dissertation explores two fundamental paradigms in enterprise integration: event-driven architecture (EDA) and managed file transfer (MFT), both analyzed under the lens of decentralization and democratization of IT development within large organizations.

Event-Driven Architecture (EDA)

As organizations grow, integrating diverse technical systems becomes increasingly complex, often requiring centralized teams to manage data flow, governance, and standardization across departments. Traditionally, this integration has been managed by a Center of Excellence (CoE), which ensures strict governance but can create bottlenecks that slow down responsiveness to business needs.

In contrast, the Center for Enablement (C4E) model shifts resource management responsibilities to individual Line of Business (LoB) teams, enabling them to manage their resources autonomously while still following organizational governance standards. Event-driven architectures, especially those using platforms like Apache Kafka, support this shift by enabling asynchronous communication and automation across systems, allowing LoB teams to produce and consume events independently.

This research focuses on transitioning from a CoE to a C4E model within both event-driven platforms and file-based integration solutions. By automating resource management and validation processes, this study aims to demonstrate how decentralizing data flow creation can reduce dependencies on central teams, accelerate integration, and support more agile operations across the organization.

Managed File Transfer (MFT)

In parallel with the adoption of event-driven systems, file-based integration remains a vital part of enterprise architecture, especially for organizations that interact with legacy systems, external partners, or processes that require batch-oriented data exchange. Despite the rise of APIs and real-time data streaming, file transfers continue to account for a significant portion of inter-system communication in many industries, including retail, logistics, and finance.

Managed File Transfer (MFT) refers to the secure and reliable transfer of files within or between organizations. Unlike basic file transfer protocols (e.g., FTP or SFTP), MFT platforms provide additional features such as automation, scheduling, pooling, tracking, auditing, encryption, and role-based access control. These features make MFT a preferred choice for enterprises that require traceability, compliance, and security in their data exchange operations (Hyvönen et al., 2018; Krishnan et al., 2022).

Historically, MFT processes have been orchestrated through Service-Oriented Architectures (SOA) or even custom scripts, typically managed by centralized IT teams. This approach often results in rigid workflows, operational delays, and a high dependency on specialized personnel for routine tasks such as configuring endpoints, setting file triggers, or managing credentials.

The democratization of MFT proposes a shift towards a self-service model, where LoB teams can independently request, configure, and manage file transfers through intuitive interfaces and automated validation pipelines (Chukhajyan, 2016; Folds & McDermott, 2019). This transition mirrors the principles of the C4E model and introduces key benefits, including:

- **Faster onboarding of file flows**: Reducing lead time from request to execution and enabling quicker integration cycles.
- **Operational scalability**: Minimizing reliance on central teams by distributing control of file transfer workflows to LoB teams.
- **Built-in governance**: Enforcing standards, encryption policies, and naming conventions through automated rule-based validation mechanisms (Folds & McDermott, 2019).
- **Auditability and traceability**: Ensuring all file movements are logged, versioned, and easily accessible for compliance (Hyvönen et al., 2018).
- **Security and access control**: Enhancing secure communication channels and access policies to meet enterprise-grade standards.

In some contexts, GitOps-inspired workflows have been used to operationalize MFT configurations, allowing for declarative specifications, automated testing, and version-controlled infrastructure (Chukhajyan, 2016). These practices align well with modern DevOps methodologies, improving visibility, rollback capabilities, and collaboration among teams.

The reviewed literature shows that MFT is not just a legacy technology, but a domain that is actively evolving to support modern integration strategies. Its alignment with automation, governance, and decentralization makes it an essential component of hybrid integration platforms, complementing event-driven solutions and enabling end-to-end data flow control across enterprise systems.

A.2 Search Strategy and Process

The search was conducted across Scopus, IEEE Xplore, and the ACM Digital Library. These databases were selected based on their comprehensive computer science and software engineering literature coverage, particularly on topics central to EDA, file-based integration, and IT governance. Five main search queries were formulated to capture relevant studies:

- **QS1**: ("system integration" OR "System integration engineering" OR "software engineering") AND ("event-driven architecture" OR "Event-based architecture" OR "Event-oriented architecture")
- **QS2**: ("Event-driven architecture" OR "EDA") AND ("kafka")
- **QS3**: ("event-driven architecture" AND "automation")
- **QS4**: ("decentralized IT" OR "decentralized model") AND ("governance" OR "IT management")
- **QS5**: ("managed file transfer" OR "MFT" OR "file transfer") AND ("integration" OR "data exchange" OR "middleware" OR "enterprise integration") AND ("self-service" OR "decentralized" OR "automation" OR "governance")

In this study, the queries were iteratively refined to obtain relevant articles, while filtering out unrelated material and minimizing the risk of overlooking significant research contributions.

To ensure compatibility of scope in all databases, and to concentrate on current research, strict search criteria were specified:

- **Scopus**: It only includes publications from 2007 forward, in English, and for articles, conference papers, and books covering within the 'Computer Science' subject area.
- **IEEE Xplore**: Only concerns studies published from 2007 onward, working with journal articles, conference papers, and books in the Event-driven Architecture subject area and restricting the included documents to those that explicitly describe Eventing concepts, operations, and native implementations.

- **ACM Digital Library**: Only referring to conference papers, articles, and books in "Computer Science", published in English since 2007.

These criteria allowed me to have some consistency in the scope of the searches to get a balanced right and recent literature for this study.

The results obtained from the search queries across the three databases were as follows:

Database	SCOPUS	IEEE	ACM	Total
QS1	23	21	11	55
QS2	10	8	6	24
QS3	24	15	8	47
QS4	13	11	15	39
QS5	10	23	24	57
Total	80	78	64	222

Table A.1: Search results from each database

For instance, this search results show the different levels of coverage on different topics, with SCOPUS returning most results than any other topics that are the amount of literature on event driven architecture, automation and decentralized IT governance.

The results obtained duplicates removal:

Database	SCOPUS	IEEE	ACM	Total
QS1	17	19	7	43
QS2	2	7	5	14
QS3	17	15	8	40
QS4	10	11	15	36
QS5	8	23	24	55
Total	54	75	59	188

Table A.2: Search results after duplicates removal

A.3 Screening and Data Collection

The Screening Phase involved a detailed review of the initial set of studies to ensure they fulfilled certain inclusion criteria so they could be used. In this phase I attempted to refine the pool of studies by removing studies that were outside the scope of objectives.

During this phase, a data extraction process was employed to compile critical information from each selected study, such as:

- **Title, Author(s), and Year**
- **Type of Publication**: Differentiating journal articles, conference papers, and books.
- **Source**: Database origin (Scopus, IEEE, or ACM).

- **Relevance**: Level of relevance to core research topics like EDA, Kafka, and IT governance.
- **Contribution**: Key insights or contributions to understanding automation, integration, or governance.

It provided a more systematic review and synthesis of findings so that only relevant to the study's research question and objectives literature was included when reading all the included literature.

Therefore, a list of inclusion and exclusion criteria was defined to determine which studies could contribute substantively towards event driven architectures and decentralized IT models.

Inclusion Criteria:

- Studies published from 2007 onwards in the field of "Computer Science" or related areas.
- Availability of full-text access.
- Articles, conference papers, or books in English.

Exclusion Criteria:

- Duplicate records across the databases.
- Works classified as "work-in-progress" or summaries from workshops.
- Studies not directly related to EDA, automation, IT governance models or MFT.

From an initial pool of 222 retrieved studies, a systematic screening process was conducted based on predefined exclusion criteria. This manual review focused on identifying studies with direct and substantial contributions to the research themes, rather than those offering only peripheral mentions. As a result, the dataset was narrowed down to 64 high-value publications.

The final set includes works that explicitly propose or evaluate frameworks, architectures, or case studies in the domains of event-driven architectures (EDA), Kafka-based data streaming, managed file transfer (MFT), automation of integration pipelines, and decentralized IT governance models. These studies form the theoretical and technical foundation for the subsequent analysis and implementation described in this dissertation.

All the selected references used to support this dissertation are cited throughout the document and listed in full in the References section.

The following graph shows the entire information collection process:

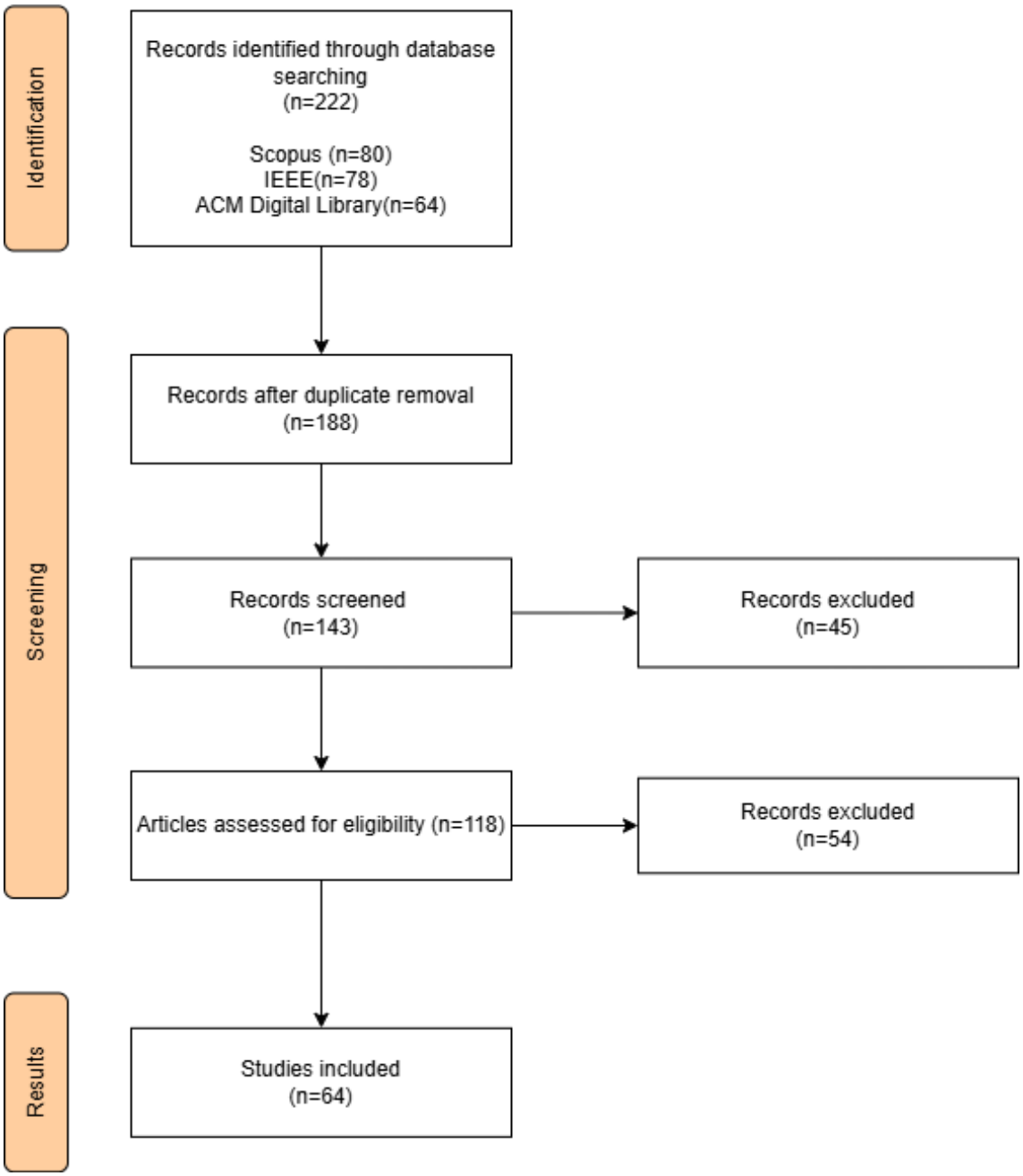


Figure A.1: Collection Process