CODICE

EMPOWERING YOUR DIGITAL JOURNEY

Transforming Legacy Systems Using Microservices

Introduction

CODICE, a Washington, D.C.-based technology services company established in 2009, helps clients transform their legacy systems. To help accomplish this, we have adopted a Microservices-based approach that breaks down monolithic structures into modular components. This approach enables our clients to modernize their systems while maintaining operational continuity.

Our legacy system transformation track record includes a variety of successful implementations, ranging from modernizing permitting systems to complex Oracle migrations and Salesforce implementations. CODICE's approach leverages cloud-native architectures, DevOps practices, and emerging technologies like AI and ML to deliver scalable, flexible solutions that help organizations enhance agility and competitiveness.

As a certified small and minority business enterprise with GSA Multiple Award Schedule certification—see the last page of this white paper for contact information—CODICE offers a complete range of services including custom software development, data analytics, cloud migration, IT security, and strategic consulting. We help both public and private sector organizations successfully navigate their digital transformation journeys.

Common Transformation Pitfalls

Transforming legacy systems offers significant benefits. It also requires careful navigation to avoid common pitfalls such as:

1. **Underestimating the complexity of existing systems**. Intricate process and system dependencies and interconnections can complicate migration to modern platforms. Conducting a comprehensive analysis of the legacy architecture is essential for developing an effective migration strategy.
2. **Neglecting change management** is another critical risk. Transformation initiatives can require business process changes and even cultural shifts. Organizations must prioritize effective communication, training, and stakeholder engagement to address employee concerns and foster collaboration.
3. **Pursuing an overly complex "big bang" migration approach** can lead to operational disruptions and downtime. Adopting an incremental, phased strategy allows for continuous feedback, risk control, and early wins throughout the transformation journey.

The CODICE Approach

Such pitfalls can be avoided. We have found that the potential of legacy system transformation can be achieved through a Microservices-based approach that breaks down complex, monolithic structures into modular components. This approach can facilitate integration of new applications, services, and functionalities while avoiding disruption of existing infrastructures.

By leveraging Microservices—increasingly regarded as the building blocks of modern software development—CODICE helps clients adapt to changing market demands, to drive innovation, and to remain competitive.

To support transformation of legacy systems, we employ a variety of methods:

- **Cloud-native architectures that** enhance scalability, flexibility, and cost-efficiency by leveraging on-demand resources.
- **DevOps practices that** incorporate automated testing and continuous integration to streamline workflows, improve reliability, and accelerate time-to-market.
- **AI and ML methods** to further optimize system performance by analysing Microservices data for predictive insights and proactive issue resolution.

Our project management approach adapts the following basic elements to individual clients:

1. Planning & Design

The initial phase of transformation that employs a Microservices architecture involves a thorough understanding of the application's needs. This begins with clearly defining business goals and functionalities. Subsequently, Domain-Driven Design (DDD) principles are applied to model the application's domain into distinct bounded contexts. These serve as the foundation for identifying and defining individual microservices. Each key business capability is then decomposed into independent fine-grained services, with clearly defined APIs utilizing RESTful principles and appropriate data formats such as JSON. Finally, the appropriate technology stack for each service is carefully selected, considering factors such as programming languages, frameworks, databases, and deployment platforms.

2. Development & Deployment

To foster independent development and ownership, dedicated teams are formed for each microservice. Each microservice should then be containerized using technologies like Docker to ensure consistent deployment and portability across different environments. Robust CI/CD pipelines should be implemented to automate the build, testing, and deployment processes. Furthermore, container orchestration platforms such as Kubernetes or Docker Swarm are essential for managing the deployment, scaling, and networking of these microservices effectively. Finally, implementing service discovery mechanisms, such as those provided by service mesh tools like Consul or Istio, is critical for enabling dynamic service location and communication within the microservices architecture.

3. *Communication & Integration*

To effectively manage inter-service communication, an API Gateway should be implemented to handle request routing, enforce security measures, and manage traffic flow through rate limiting for both external and internal requests. Asynchronous communication patterns, such as those enabled by message queues like Kafka or RabbitMQ, are crucial to decouple services, enhancing scalability and resilience by minimizing dependencies and improving fault tolerance. To maintain data consistency across the distributed system, strategies such as event sourcing or utilizing a distributed database should be employed.

4. *Modular Architecture*

A modular approach to system transformation using Microservices offers these advantages:

- **Flexibility Through Modularity**: CODICE enables businesses to break down monolithic applications into smaller, manageable Microservices.
- **Simplified Maintenance**: The modular architecture simplifies development and maintenance while allowing independent updates and new feature deployments.
- **Increased Agility**: Organizations gain greater responsiveness to market changes and minimize downtime.

5. *Security*

Employing Microservices provides security related advantages:

- **Prioritized Security Measures**: CODICE includes advanced features like encryption, access controls, and threat monitoring to safeguard data.
- **Compliance Support**: Organizations can ensure regulatory adherence throughout the transformation process.
- **Confidence in Transformation**: CODICE mitigates security risks, enabling organizations to transform with assurance.

6. *Cross-Functional Teamwork*

- **Establishing Teams**: A cross-functional team with representatives from IT, business, and compliance ensures diverse perspectives.
- **Collaborative Execution**: Stakeholders collaborate to identify challenges and align goals for effective transformation.
- **Gradual Implementation**: Using Microservices, organizations can decompose legacy components and integrate new functionalities for desired outcomes.

Successful Transformation Projects

The following are examples of successful CODICE transformation projects:

- **Modernized Permitting System:** The project developed a cloud-native permitting system, aligning with the principles of Microservices to allow modular functionalities such as submission, validation, approvals, and compliance checks, which can be independently managed and scaled.
- **Benchmarking Data:** CODICE optimized and enhanced the client's energy benchmarking website by consolidating legacy map data structures and integrating modern web and mobile capabilities for better data access and usability.

- **Salesforce Architects:** CODICE migrated the client's legacy systems to the Salesforce Lightning framework, enabling enhanced digital modernization and enterprise data management. This involved designing and maintaining systems using Salesforce Lightning, which supports modular and reusable components.
- **Data Lake Project:** CODICE implemented robust data pipelines for integrating and analysing large datasets from multiple sources. The use of Microsoft Data Factory for automated processing implies modular integration components.
- **Oracle Migration:** CODICE transitioned the client's Oracle Cloud HCM system and integrated it with other platforms. The project eliminated duplicate data entry and modernized planning and budgeting processes using Oracle EPM Cloud.
- **Oracle EPBCS Cloud Migration:** CODICE migrated the client's Oracle Hyperion Planning applications from on-premises to Oracle's Enterprise Planning and Budgeting Cloud Service (EPBCS), modernizing planning and budgeting functionalities. The involved modular architecture, including separating HR-related and project-related functionalities into distinct plan types.
- **Child Support Enforcement System:** CODICE modernized human services systems employing cloud-native architecture and advanced DevOps frameworks to replace older systems with more efficient, scalable solutions.
- **IT Consultation and Salesforce Implementation:** The architecture and engineering tasks for cloud-native environments, including PaaS (Salesforce) and Kubernetes (for orchestration), suggest the use of Microservices for scalability, flexibility, and seamless integration with various components.

Let's Talk!

Contact:	Dash T. Kiridena, CEO, (202) 779-5400, dash.kiridena@codicetech.com
Company Name:	Health IT 2 Business Solutions, LLC, dba **CODICE**
Website:	www.codicetech.com
Location:	1101 Vermont Avenue, NW, Suite 400, Washington, DC 20005
Categories:	Certified Small and Minority Business Enterprise; DOT Certified DBE; Subcontinent Asian (Asian Indian) American Owned.
GSA Multiple Award Schedule (MAS):	Contract Number GS-35F-157GA (special item number 54151S)
Unique Entity ID:	GTX2JFCTPNW5
CAGE Code:	5SY61
NAICS Codes:	541511, 541512, 541513, 541519, 541611, 541690, 541720, 541990

In the above text, several mentioned companies and products are trademarked by their respective owners including Oracle, Docker, Kubernetes, Consul, Istio, Kafka, RabbitMQ, Salesforce, and Microsoft.