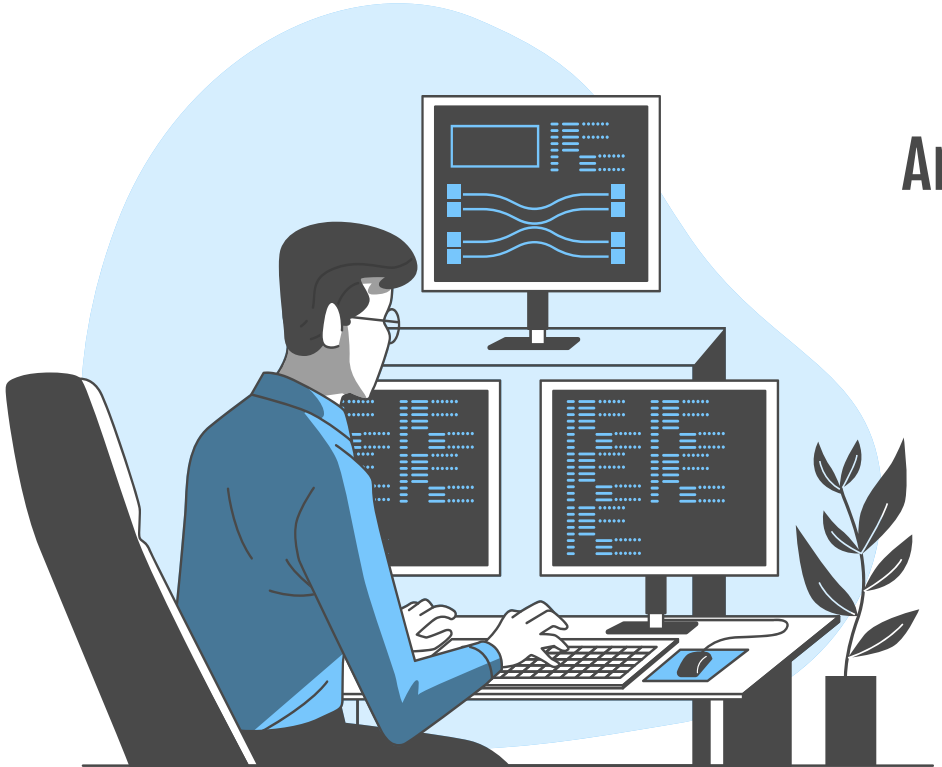

Evolution of Enterprise System Architecture: Traditional Application Development and Enterprise Application and their Integration

Renz Allen Miller
Clouie Ann Aclinen
Ika Licudan
Shina Marie Apolinar

Enterprise System Architecture (ESA)

- It is the overall IT system architecture of an organization.
- This architecture is the key part of managing and evolving IT systems, and therefore the business operations of an organization.
- It consists of the architectures of individual systems and their relationships in the perspective of an organization

Traditional Application Development

- Traditional application development is an older method of creating apps now referred to as legacy applications.
- They operate in a client-server or mainframe environment and have a monolithic architecture.

Enterprise Application

- Large-scale software solutions designed to streamline and automate various processes of an organization's operations.
- These solutions are intended to increase productivity, efficiency and collaboration across departments.
- Enterprise applications are sophisticated, comprehensive software solutions designed for large organizations (enterprises).

Evolution of Enterprise System Architecture

Traditional Application Development – (1960s–1980s)

- Applications were standalone, developed to serve individual departments (e.g., payroll, inventory).
- Developed using monolithic architectures on mainframes.
- Limited data sharing, resulting in redundancy and inefficiency.
- No integration between systems.
- Applications were data-isolated (each system had its own database or file).
- Business processes were manual or semi-automated.
- Difficult and costly to update or maintain.

Evolution of Enterprise System Architecture

Departmental Client-Server Systems – (1980s–1990s)

- Introduction of PCs and LANs allowed client-server applications.
- Slight improvement in data sharing across departments.
- Systems still siloed, with custom point-to-point integrations
- Some departments shared databases via LANs.
- Improved usability and efficiency but lacked full enterprise-wide integration.

Evolution of Enterprise System Architecture

Enterprise Resource Planning (ERP) - (1990s–2000s)

- Emergence of ERP systems like SAP, Oracle, and PeopleSoft.
- Unified system architecture covering HR, Finance, Supply Chain, CRM, etc.
- All core business functions integrated into a single software platform.
- Database centralization improved consistency and real-time data access.
- High cost and complexity of implementation.

Evolution of Enterprise System Architecture

Integration of Legacy & Enterprise Applications - (2000s)

- Organizations had a mix of new ERP systems and older legacy systems that couldn't be discarded easily.

Technologies used:

- Middleware (e.g., IBM WebSphere)
- Enterprise Service Bus (ESB)
- Message Queuing (e.g., JMS, MQ Series)
- XML, SOAP, WSDL for web services
- Adapters to "wrap" legacy systems

Evolution of Enterprise System Architecture

Integration of Legacy & Enterprise Applications - (2000s)

Results:

- Seamless communication between old and new systems.
- Real-time or near-real-time synchronization.
- Reuse of legacy logic without rewriting.

Evolution of Enterprise System Architecture

Service-Oriented Architecture (SOA) - (2010s)

- Applications broken into services for reusability.
- Web services (SOAP, WSDL) enabled cross-platform integration.
- Allowed businesses to be more agile and flexible.

Evolution of Enterprise System Architecture

Microservices and Cloud-Native Architectures – (2015–Present)

- Systems designed as collections of small, loosely coupled services.
- Use of RESTful APIs, containers (Docker), and orchestration.
- Scalability, fault tolerance, and DevOps practices integrated.
- Shift to cloud platforms (AWS, Azure) for flexibility and global access.

Evolution of Enterprise System Architecture

Event-Driven & AI-Enabled Enterprise Systems - (Present–Future)

- Emphasis on real-time data streaming.
- Integration of AI/ML into business workflows.
- Move toward Composable Architecture using APIs, and low-code/no-code tools.

Thanks!