Retire the Three-Tier Application Architecture to Move Toward Digital Business

Published: 21 June 2016

Analyst(s): Anne Thomas, Aashish Gupta

The three-tier application architecture is obsolete and no longer meets the needs of modern applications. CTOs should direct their architecture teams to adopt the mesh app and service architecture (MASA) to support their digital transformation initiatives.

Key Findings

- Now 20 years old, the three-tier architecture was designed to support the application requirements prior to the explosion of web and mobile apps. Application requirements have changed drastically since then, and three-tier is woefully inadequate to meet them because of its rigid and linear design.

- The mesh app and service architecture (MASA) is the preferred application architecture for the digital age. MASA applications natively support optimal, multichannel user experiences, and their agile architecture enables continuous delivery of new capabilities. They can support diverse data sources, leverage pertinent context information for better automated decisions, and exploit the API economy and the external algorithm marketplace.

- The distributed nature of MASA can generate increased operational complexity, but it improves agility and scalability, which are primal requirements for any digital business.

Recommendations

For CTOs:

- Set up an innovation program for experimenting and building expertise with MASA. Execute pilots for both new application development and monolithic application refactoring.

- Task your architects with defining technical architectures, standards, governance mechanisms and success metrics for MASA.

- Develop a roadmap for your digital transformation journey. Assess your application portfolio and identify applications that are critical to that journey. Build a business case for rearchitecting these applications to MASA.
Table of Contents

Analysis..................................................................................................................................................2

What You Need to Know.................................................................................................................. 2

Overview.......................................................................................................................................... 3

Modern Application Requirements....................................................................................................5

The Changing Front End.....................................................................................................................8

Optimized User Experience..............................................................................................................8

Diverse Data Sources and Persistence Models................................................................................11

Continuous Delivery of New Features..............................................................................................12

Injecting External Context for Better Decisions.............................................................................13

Transitioning From Three-Tier to MASA...........................................................................................15

Gartner Recommended Reading..........................................................................................................16

List of Figures

Figure 1. The Three-Tier Architecture......................................................................................................4

Figure 2. The Mesh App and Service Architecture.................................................................................. 5

Figure 3. The Changing Nature of User Experiences...............................................................................7

Figure 4. Modern Applications Support Multiple Client Devices and Channels........................................ 8

Figure 5. Distinct Apps for Different Personas....................................................................................... 10

Figure 6. Modern Applications Support Multiple Data Sources and Persistence Models....................... 12

Figure 7. Microservices and Miniservices Enable Continuous Delivery..................................................13

Figure 8. Applications Should Leverage External Context..................................................................... 14

Analysis

What You Need to Know

Three-tier architecture, typically implemented using frameworks such as ASP.NET and Java EE, has been the predominant application architecture for the past decade. These architecture frameworks emerged in the late 1990s as a means to deliver web applications. But digital business demands an application architecture strategy that supports more than just a web front end. Consumers now demand access through multiple channels: web apps, mobile apps, social media, notifications, email and more. Competitive pressures are driving organizations to provide user experiences that exceed expectations, and leading digital business players ensure that users have optimized, continuous and ambient experiences as they traverse these channels. And the number of possible endpoints in the device mesh continues to expand (see Note 1). Meanwhile, organizations will fall
behind if they aren’t prepared to leverage third-party algorithms and data sources, or rapidly deliver new features.

Three-tier architecture isn’t designed to support these requirements. Three-tier presumes a monolithic, linear model in which an application supports one primary client interface: a browser. It doesn’t support optimized experiences for different personas or different channels. The business logic tier is tightly bound to the user interface (UI) tier, and the data tier is tightly bound to the business logic tier. These tight bindings produce a static and brittle monolith that impedes agility and makes it difficult to share functionality or data. The architecture presumes that all business logic and data are owned and managed by the one application, and it doesn’t natively accommodate third-party algorithmic services or diverse, distributed data sources.

Modern applications require an architectural model that mimics the dynamism of the digital mesh (see Note 2): a design that promotes autonomy and collaboration in problem solving and that natively supports an ever-expanding set of interface channels, services and data sources. A new architectural model has emerged that addresses these modern application requirements: The mesh app and service architecture (MASA) is a modular architecture born of the digital business age. MASA comprises multiple independent and purposeful modules that can be composed as needed to support multiple channels, users, roles and networks for delivering application functionality.

Overview

Three-tier (or n-tier) architecture logically divides an application into three or more tiers. The top tier manages user interaction, the middle tier(s) implement the business logic, and the bottom tier manages data access. It’s now time to move away from this model.

The problem with the tiered approach to application architecture is that it defines applications in only one dimension (see Figure 1). The UI (a browser) is at the top; the data source (usually a relational database) is at the bottom. The business logic in between is tightly bound to the UI and the database. The entire application is dedicated to supporting one monolithic experience. It connects the UI to the database. Top down. Bottom up. Straight line. One dimensional. One size fits all.
This is not the way applications should be built today. We can no longer accurately depict application flow as top-down, and then bottom-up. Instead, applications must support multiple, distinct application experiences, accessed through multiple client channels. Business logic must support multiple workflows to support those distinct experiences.

Often, an application incorporates functionality from other applications and shares this with external systems. Data typically comes from multiple sources, gets pushed in multiple directions and gets stored in diverse formats and models. Data sources are typically abstracted and encapsulated by the services that manage them. Rather than a simple linear model, a modern application architecture diagram looks like an interconnected mesh of independent apps and services (see Figure 2). Individual services may be deployed on-premises or in the cloud, and sometimes accessed from third-party sources. An app or service can participate in multiple applications. Often, an application consuming a service does not "own" it. See “Top 10 Strategic Technology Trends for 2016: Mesh App and Service Architecture" for an overview of this new architecture.
Modern Application Requirements

We are moving toward a world where applications and their consumers will be omnipresent. This has already begun with "things" rapidly getting entwined in the virtual labyrinth. Users want access to functionality to perform their tasks from anything and anywhere: at home or in the office; at the game or in a restaurant; from the car or a plane (see Figure 3). In order for a business to capitalize on this digitalization, it needs to provide interfaces at all the relevant consumer touchpoints. This implies that a business will be truly digital only if it is able to extend its perception of potential customers from people to things (see "Digital Business Is Here Now"). Connecting disparate things may involve heterogeneous technologies. This calls for an architecture that has flexible implementation and integration capabilities — not for something as rigid as a monolith.

More channels shouldn't lead to more applications. Owning and maintaining separate applications for each channel with an exhaustive set of features would be impractical (see "It's Time to Gradually Give Up Developing, Buying and Maintaining Applications"). Instead, organizations should design small, purposeful apps for the different channels, and ensure that those apps can share a common set of back-end services. MASA enables this by leveraging APIs and reusable services. These
services can be managed in-house, or applications can leverage third-party services managed by external service providers. Capabilities within an application can be exposed via APIs for use by other applications. In some cases, those capabilities could be exposed for consumption by third parties, enabling the organization to participate in the API economy — see "The API Economy: Turning Your Business Into a Platform (or Your Platform Into a Business)." Although three-tier architecture does technically support shared services, its tightly coupled nature does not facilitate sharing. MASA, on the other hand, promotes sharing by encapsulating individual functionalities behind well-defined interfaces, making each of them a possible candidate for reuse.

This service-oriented approach decouples layers and functionalities of software architecture to:

- Allow the front end and back end to change
- Provide a more optimized user experience
- Diversify data sources
- Ensure continuous delivery of applications
- Help in algorithmic decision making
The Changing Front End

Consider just the front end of a modern application. Applications may need to support a broad range of UIs running on an array of different client devices, including PCs, tablets, smartphones, watches, kiosks, car dashboards, GPS devices and media players. They may need to support multiple logical channels, including browsers, apps, social networks, digital personal assistants (DPAs), web robots (bots) and other services. They may need to accept requests from IoT devices and automated processes. The list is long and continues to grow (see Figure 4). Many front-end clients will come from outside your organization. In this context, the notion that an application’s business logic must support only one client interface is clearly quaint.

Figure 4. Modern Applications Support Multiple Client Devices and Channels

DPA = digital personal assistant; Bot = web robot; IoT = Internet of Things

Source: Gartner (June 2016)

Optimized User Experience

In the traditional, three-tier architecture, the number of capabilities offered by the application is considered to be a key metric for determining its value. As a result, users are inundated with features they never use. Unlike "application," "app" as a concept derives its value from how well it serves a specific purpose (see "The App and Its Impact on Software Design").
An app provides a specific UX for an application. It provides a fit-for-purpose experience for a particular persona executing a specific task using a particular channel (see Figure 5). An app is designed for simplicity and intuitiveness. Contrast that to a typical three-tier application, which is designed to support all personas and all tasks from one interface. Therefore, an optimized system will have multiple apps, each catering to a defined user role — an idea contrary to the "one application supports one client interface" design propagated by many implementations of the three-tier architecture.
Figure 5. Distinct Apps for Different Personas

My Travel Requests

Approve Travel Requests

An Employee Task

A Management Task

Source: Gartner (June 2016)
Modern applications further enhance the user experience by providing a seamless transition of workflow as the user moves from one device or channel to the other. Channels include both digital and analog, such as mobile devices, web, social media, IVR, SMS, email, print, DPAs, call centers and in-person channels. Businesses can use this continuous experience approach to increase customer satisfaction, reduce customer attrition and improve employee productivity. For example, a utility customer with a service outage often uses multiple channels to report the problem and request status updates. That customer (who is most likely aggravated by the outage) would be further aggravated if she needs to repeat the context of the outage for each channel. On the other hand, she will be more easily placated if each interaction channel has full knowledge of all her previous interactions related to the outage. Implementation of continuous experience requires collaborative back-end application services (instead of monoliths working in silos) and shared data models to capture state, to enable users to shift across interaction models.

The "app" and "continuous experience" concepts underline the idea of "digital humanism," which promotes people-centered over process-centered designs (see "How to Apply Gartner's Digital Humanism Manifesto"). In times ahead, when technology will become overwhelming, when its complexity will transcend our understanding, a human-centric approach will be the key differentiator and source of resilience for digital businesses.

Diverse Data Sources and Persistence Models

The back end of a modern application is often quite different, too (see Figure 6). Many modern applications employ multiple data sources and persistence models, all of which may be controlled by different parts of the enterprise or even different enterprises. In many situations, a service encapsulates and manages its own data, and it uses a data persistence model that best suits the service’s requirements. But this results in applications that rely on multiple data storage and data access mechanisms:

- Relational databases
- NoSQL and big data
- Files
- Feeds
- Event streams
- Other applications
Continuous Delivery of New Features

Digital business requires rapid response to opportunities: Application architecture must be agile enough to enable the continuous delivery (CD — see Note 3) of new features. Monolithic architectures impede agility: Every new feature must be integrated into the next release of the monolith, and for most organizations, release delivery schedules are measured in months, at best. Microservices architecture (and its more relaxed and less disruptive sister, miniservices architecture) decompose monolithic application logic into independently deployable services that can be deployed as soon as they are completed (see "Innovation Insight for Microservices" and "Assessing Microservices for Cloud-Native Application Architecture and Delivery").

In a typical monolith, deployment of new features poses a high risk of making breaking changes due to its tightly coupled modules. The risk increases as these features accumulate over time to be deployed en masse in the next release cycle, making it difficult to isolate the breaking change in case of a failure. Moreover, this impedes innovation as development teams become disinclined to come up with breakthrough features. On the other hand, in a design using independently deployable services, the risk of making breaking changes is reduced as services are loosely coupled and small (see Figure 7). The team owning a service enjoys autonomy and flexibility in pushing changes or new features into production (with necessary versioning and backward compatibility in order to avoid violation of service contract). Also, these services are independently scalable, allowing DevOps to scale just the part of the application that is causing the bottleneck instead of deploying multiple instances of the entire application.

Agility and CD have been the primary motivations for organizations to invest in microservices architecture. Implementing this architecture can increase the application's complexity, and it
requires associated investment in new infrastructures and processes; but, at scale, the associated benefits far outweigh the challenges. Businesses employing archaic architecture run the risk of product obsolescence even before the new version is released. On the other hand, delivering software and services more quickly can translate into new business and higher market share. In order to stay competitive, organizations should embrace CD methodologies and automate a CD pipeline (build, test, package and deploy — see "Blueprint for Architecting a DevOps Continuous Delivery Pipeline"). Container-based virtualization technology (such as Docker) can facilitate CD practices and microservices adoption (see "Become More Agile and Get Ready for DevOps by Using Docker in Your Continuous Integration Environments").

Figure 7. Microservices and Miniservices Enable Continuous Delivery

Injecting External Context for Better Decisions

In a growing number of circumstances, digital business opportunities move too fast to wait for humans to make decisions. These opportunities require organizations to automate more decisions. As these automated decisions become more critical to the business, the algorithms making the decisions must become more sophisticated by leveraging additional context.

Context data can come from internal historical data sources (for example, ODSs and MDM stores), although more-intelligent decisions may also require external context data (see Figure 8). This external data emanates from context data brokers, devices streaming real-time data and external applications. Due to growth in digitalization, there is no dearth of external context data and, with advancement in analytics, the cost of sourcing it is also decreasing. But the task of sourcing and injecting the humungous amount of context data will become overwhelming if its logic resides inside
the application code itself. This is contrary to a monolithic design where all the code resides in one application. Therefore, a service-oriented architecture is more adept at handling such complexity, allocating these responsibilities to separate services that feed events to decision logic (inside the application) through an API (see "Architecture Strategy Guide for Context-Aware Algorithmic Decisions" for more information).

Figure 8. Applications Should Leverage External Context

IoT = Internet of Things; MDM = master data management; ODS = operational data store

Source: Gartner (June 2016)
Transitioning From Three-Tier to MASA

For most organizations, the migration from three-tier to MASA will be gradual rather than a "big bang" transition. Many applications can remain as they are, but your digital business initiatives will require some to evolve to MASA. Certainly, your new applications should be built using MASA. This transition demands both cultural and technological shifts. Here are a few steps that can help you evolve your application architecture and become "digital-business-ready."

- Set up an innovation program that allows your architecture teams to experiment with MASA and build expertise. Start with an existing three-tier application and refactor it into apps and services. Ask your enterprise architects to identify a low-risk application for the task to ensure that unwanted changes do not hamper your critical processes. Also, identify a "greenfield" opportunity to learn about building new applications using MASA.

- Invest in research and training to develop necessary skill sets and deploy new infrastructure to support CD automation and microservices and/or miniservices runtimes. Consider using open-source technologies from companies that have successfully implemented MASA-based architectures.

- In order to fully understand the viability and impact of the new architecture, perform experiments at both service and operation levels. Service-level tests may involve breaking the existing modules from the monolith in piecemeal fashion (and then connecting them via well-defined APIs), or keeping the monolith intact and building new functionalities in small, tightly scoped services around it. Operation-level tests will require implementation of tools that manage service life cycle, monitoring and communication. The challenges and impact that you observe in this process will help you assess your readiness for MASA and identify areas where you need to improve.

- As you build expertise, task your architects with defining technical architectures, standards, governance mechanisms and success metrics for MASA, to enable more-consistent adoption of the architecture.

- Develop a roadmap for your digital transformation journey. Assess your application portfolio and identify applications that are critical to that journey. Build a business case for rearchitecting these applications to MASA.

Acronym Key and Glossary Terms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>continuous delivery</td>
</tr>
<tr>
<td>DPA</td>
<td>digital private assistant</td>
</tr>
<tr>
<td>IVR</td>
<td>interactive voice response</td>
</tr>
<tr>
<td>MDM</td>
<td>master data management</td>
</tr>
<tr>
<td>ODS</td>
<td>operational data store</td>
</tr>
</tbody>
</table>
Gartner Recommended Reading

Some documents may not be available as part of your current Gartner subscription.

"Top 10 Strategic Technology Trends for 2016"

"Top 10 Strategic Technology Trends for 2016: The Device Mesh"

"Top 10 Strategic Technology Trends for 2016: Mesh App and Service Architecture"

"Digital Business Transformation: Turning the Digital Dream Into Reality"

"The Platform Architect's Guide to Designing IoT Solutions"

"Innovation Insight for Microservices"

**Note 1 Device Mesh**

The device mesh refers to an expanding set of endpoints people use to access applications and information, or to interact with people, social communities, governments and businesses. The device mesh includes mobile devices; wearable, consumer and home electronic devices; automotive devices; and environmental devices — such as sensors in the Internet of Things (IoT).

**Note 2 Digital Mesh**

The digital mesh is the collection of devices (including "things" from the IoT), information, apps, services, businesses and other people that exist around an individual. As the mesh evolves, all devices, compute and information resources, businesses, and individuals will be interconnected. The interconnections are dynamic and flexible, changing throughout the day.

**Note 3 Continuous Delivery**

Continuous delivery (CD) is an advanced DevOps practice that automates the process of building, testing, packaging and deploying code. For more information on CD, see "Market Trends: DevOps — Not a Market, but a Tool-Centric Philosophy That Supports a Continuous Delivery Value Chain" and "Blueprint for Architecting a DevOps Continuous Delivery Pipeline."