IDC
Analyze the Future

IDC TECHNOLOGY SPOTLIGHT

From Macrosystems to Microservices: Lightweight Development with the Oracle Cloud

February 2016

Adapted from *The Emergence of Microservices as a New Architectural Approach to Building New Software Systems* by Al Hilwa, IDC #256906

Sponsored by Oracle

It has been said and repeated that software is eating the world. The heightened sense of industry disruption taking place today is largely enabled by software. A palpable shift in the understanding of the importance of software in recent years is rapidly morphing into a scramble to build software development competencies as enterprises cope with escalating disruption. But software is not easy. Software development is an intense discipline requiring deep and layered investment in technical skills. Software development is hard to learn and an order of magnitude or two harder to master. Software quality, productivity, and project success have persistently remained elusive. In this context, modern enterprises focusing on digital disruption cannot afford not to leverage every advancement in the science and practice of software development. Microservices architectures, lightweight and script-based applications, and cloud-native application development join the ranks of agile in moving the bar forward in the endeavor to build better software faster. This paper explores evolving practices in successful software development, highlighting the benefits of the move to cloud architectures.

Introduction

Modern enterprises now realize that they need to continually innovate and transform. Enterprises are also discovering that they need new software development competencies to carry through this transformation, unlike what they have been practicing. In the past five to ten years, we have seen a variety of new techniques make inroads into the enterprise. For example, enabled by advancement in testing automation, continuous integration is reaching mainstream adoption in enterprise development. We are also seeing increasing adoption of test-driven techniques, which prescribe that the tests be developed prior to the code to ensure complete coverage and provide a level of assurance that the requirements are captured with some fidelity. In addition, new approaches to application architecture, such as microservices, have emerged to leverage faster networks by building distributed systems with smaller code modules connected through published APIs. Finally, the expediency of modern business and the escalating need to build software faster have made it impossible to ignore the benefits of cloud computing.

A Brief History of Enterprise Software

Organizations have been building software for over a half century, and developers have always been seen as important agents of organizational change. In the early days of computer automation, companies focused on building manufacturing and distribution tracking systems. Then a broader range of enterprises turned to automating horizontal organizational functions, which run back-office or front-office systems, such as financial management, human resources, supply chain management, and sales force automation. Initially, these systems were custom implemented inside of each enterprise. Over

time, as technology markets matured and the software supply ecosystem thickened, a rich marketplace of ready-made commercial application packages evolved, thus heralding the golden age of enterprise resource planning (ERP). Such application packages enabled small to midsize companies to compete with larger companies on a more level playing field, often without any internal software development. Most enterprises were in the process of shifting to this buy-versus-build approach. For a moment, it seemed that enterprises were automating such back-office systems so rapidly that we were fast reaching saturation in automation. Then, in the mid- to late 1990s, the Internet revolution kicked in and unleashed a new wave of enterprise transformation centered around electronic commerce. The digital marketing revolution was also born in that era. Enterprise application development diversified from customizing and extending ERP to building significant new systems of engagement that targeted customers that were increasingly online.

In 2007, the iPhone was introduced and began a new shift. Mobile devices transformed a gradual shift toward more online customer interaction conducted through personal computers into the mobile customer engagement revolution. With mobile devices, customers can interact with businesses more frequently and intimately and in richer ways enabled by cameras, geolocation, and ubiquitous wireless connectivity. A frenzy of mobile storefronts and engagement apps stimulated a parallel frenzy in the use of public cloud capabilities in the form of infrastructure as a service (IaaS), platform as a service (PaaS) and, eventually, mobile backend as a service (MBaaS) offerings. Such systems started generating gushing rivers of new interaction and sensor data, fortified by location, images, and exploding content. It is expected that the proliferation of IoT devices will only multiply this data and the need for faster and more capacious back-end processing systems.

In recent years, mobility has begun to encompass the reengineering of employee apps that are often extending and layering on top of core ERP systems built before. Having taken control of optimizing internal operational efficiencies, developers have become the key enablers for business growth. Thus the modern digital transformation agenda has come full circle to embrace both customer and employee engagement and automation, fully planting mobile and cloud developers in the heart of every transforming enterprise. Supporting software development at the new pace of business has come to require new approaches, new architectures, and new infrastructure.

The Shift to Microservices and Lightweight Development

Rooted in the basic concepts of modularity and separation of concerns, the driving principles of microservices go back to the ancient history of software engineering. Indeed, software engineers have been studying, debating and, on occasion, clarifying and improving these basic ideas for the best part of the past 50 years. Recently, however, the industry has witnessed a coming together of several software development practices to define a coherent new set of practices and accompanying architectural style. Software engineering teams operating at the vanguard of digital disruption in tech companies such as Amazon, Netflix, Facebook, Uber, and eBay were practicing this approach to realize innovation at a fiercely rapid pace. The approach was labeled according to one of its characteristic features — the construction of complex systems out of as small as possible functional software subsystems, often as a network service, thus microservices. Implied in the functional unity is a level of independence in the way a microservice is constructed, which is probably an even more important characteristic: Microservices are designed to be independently evolved. Lastly, in order to provide holistic system functionality while retaining the independence of evolution, services must rely on published APIs that form the basis of the interoperability contracts between them.

It's in the API

APIs are at the center of digital disruption because by allowing cross-enterprise systems to interact, they enable fine-grained access to external customer engagement data and function to be consumed. In microservices architecture, APIs also play the central role in internal service interoperability. To support a high level of independence, successful microservices architectures require a strong API design and evolution capability to manage service interfaces. In well-functioning and more mature microservices and cloud-based organizations, API design and evolution are governed by well-defined practices and a well-functioning virtual team drawn from the various services teams. APIs are the composition language of microservices, and thus every team must respect organizational practices in API design and evolution so that the overall system functions. API change management and versioning should include central design and policy discussions inside of a service team organization, and the use of tools such as impact analysis reporting is a best practice. Also important is scalable delivery through an API management layer. API design and management skills are an essential core competency in a microservices or cloud-based initiative.

Microservices benefit from a decade-long evolution in tooling and automation as well as the thinking and rethinking of software development and deployment processes that allow developers, designers, and architects to build more effective architectures.

Teaching New Apps Old Tricks

Most enterprises have a rich application portfolio built up over a number of years, if not decades. Moreover, enterprises typically have significant heterogeneity in platform technologies, development tools, and software architectures employed. IDC estimates that at any point in time, most enterprises — even those that are engaged in a high degree of digital transformation — are actively evolving less than 25% of their total application portfolio. Even with an active set of applications, it is important to prioritize new initiatives. Modernizing existing applications is complex and varied and is heavily discussed in the industry. The following set of guidelines only scratch the surface of how the world of application modernization can intersect with the world of lightweight apps:

- The primary focus for modern application development should be new systems and projects wherever possible. The level of organizational and cultural change required for a successful microservices and cloud-oriented architecture rollout is significant, and it should be prototyped and refined by small "innovation teams" that can perform a scouting mission to specifically trailblaze how the organization and culture can be adapted to the new cloud-based world of software development. These missions should not be throwaway projects, but a high degree of risk is implied, and the culture of "fail fast and fail often" should be the mantra.
- Inevitably, existing internal systems will require integration with a new cloud–based system. In such cases — depending on the system — traditional wrapping, data interoperability, or SOA techniques can be used.

Microservice and Cloud Development Success Factors

To obtain the benefits of a microservices and cloud-based approach to software development, organizations should develop to achieve higher maturity levels along a few important dimensions. Generally, these developments can be aided by external products and services, such as tools and cloud services, or by internal efforts aimed at organizational evolution. We discuss two of each briefly in the following sections.

Tools and Automation

A high degree of automation is essential at every stage of development and deployment in the cloud. IDC believes that the continued advancement in development tools, specifically tools focused on continuous integration/continuous delivery (CI/CD) and DevOps workflows, will play a key role in maturing microservices and cloud-oriented environments. Continuous integration relies on the power of automated testing, and continuous delivery requires that a sufficient level of automation be placed to ensure that deployment can be fast and error free as well as a nonissue. A large variety of software categories, including application management tools, monitoring and log tracking and analysis tools, and project and version management tools, will conspire to improve the state of modern app development over the next five years. IDC expects activity in developer automation tools to continue on an aggressive evolution, driven by a number of players and open source projects in the industry and the increased adoption of these tools by enterprises.

Cloud Transformation

Organizations are doing more and more in the cloud, especially in PaaS and IaaS clouds such as those available from Amazon, Salesforce, and Microsoft and more recently established enterprise software players such as Oracle and IBM. Workloads in the cloud are easier to monitor and are more measurable, manageable, and elastically stackable. Cloud infrastructure, in particular, allows easier leveraging of existing systems as components of new services and can thus catalyze enterprise adoption of microservices. In addition, PaaS and IaaS offerings are increasingly adding preintegrated development tools to support modern DevOps and CI/CD workloads, thus bolstering the level of capability of modern development teams and expediting the adoption of new automation practices. API management services are also increasingly being integrated into PaaS and IaaS offerings. As cloud transformation continues and traditional enterprise players such as IBM and Oracle mature DevOps capabilities to their clouds, enterprise adoption of the practices that underlie microservices will increase.

Skill Advancement

Overall, the skill levels of today's developer community with the tools, techniques, and practices of microservices and cloud-native development are relatively low. In addition to the diversity and low maturity of tools that have to be learned and integrated to build the needed developer infrastructure, new projects often involve the use of newer programming languages such as Go or Node in which expertise is rare and newly cultivated. Developer skills around tools, especially programming languages, evolve slowly because mastering new programming language can take two to five years for the average developer. This can prove to be a significant barrier for the adoption of new architectures. For this reason, the overall industry competency with service-based architectures is expected to evolve gradually but not explosively as might be projected by media or vendor hype.

Organizational Evolution

DevOps, as a set of practices, is critical to cloud-based development. To support DevOps appropriately, developer teams essentially have to be willing to undertake end-to-end operational tasks. In larger organizations, to address more comprehensive microservices initiatives, the DevOps culture has to reach a significant chunk of the active IT portfolio of applications. IDC estimates that today, less than 10% of organizations have adopted DevOps practices to any significant degree in supporting production operations. IDC expects a significant evolution in DevOps adoption, potentially reaching the milestone of 50% of organizations utilizing DevOps practices in all of their new projects by the end of the decade. We expect that a significant chunk of these organizations will run microservice and cloud-based architectures.

Oracle Cloud — A New Approach

Company Background

Founded in 1977, Oracle (Redwood Shores, California) specializes in developing and marketing computer hardware systems and enterprise software products. Oracle gained its fame by building the most successful database product in the software industry. Starting in the 1990s, Oracle began to diversify its software holdings, transforming itself into an end-to-end supplier of enterprise software solutions spanning database, middleware, and enterprise applications. In 2010, Oracle acquired Sun Microsystems and became a high-end server system vendor and the custodian of Java, which is the core technology of its entire software stack. In the past two years Oracle has launched its most important effort to reinvent itself as a cloud provider. Oracle's strategy — beyond garnering new customers of course — is to integrate the company's new and constantly expanding set of cloud services into the IT portfolio of Oracle's 400,000+ existing customers and become a one-stop provider for their on-premises and cloud services needs.

Oracle Developer Cloud Service

Oracle Developer Cloud Service is one of the critical new services that Oracle has rolled out recently to help its customers on their transformation agenda journey. The set of services is designed for the building and delivery of modern applications that leverage modern continuous delivery workflows. The service provides the following key capabilities:

- The ability to create and manage project source files through GIT source code control system repositories and integration with the popular GitHub service
- Support for dependency management with the Java Maven build automation tool
- The ability to perform team source code reviews
- The ability to build integration using the Hudson continuous integration system
- A team wiki and a set of collaboration tools to support the documentation needs of developer projects
- IDE support with a choice of Java IDEs popular with enterprise developers including JDeveloper, Eclipse, and NetBeans
- Collaboration tools and issue tracking such as increasingly required by modern application development and DevOps practices

Oracle Developer Cloud Service offers the flexibility to bridge the gap between "old" and "new" styles of application development and deployment, with the ability to deploy apps on-premises or in the cloud in an integrated manner.

Oracle Application Container Cloud

Oracle Application Container Cloud is a new Oracle PaaS offering that takes advantage of Docker containers to enable modern DevOps workflows. The service currently supports two programming languages with plans for additional popular languages (such as PHP and Ruby along with others) on the road map. The service includes:

- Java SE Cloud Service, which provides support for modern lightweight frameworks such as Spring, Jersey, and Tomcat and can run JVM-based languages such as JRuby and Closure

- Node Cloud Service, which is optimized to run JavaScript-based server-side functionality using the popular Node JS platform with support for Node frameworks and npm modules such as Express and Passport

Challenges and Opportunities

Oracle has built a solid new offering for enterprise developers tasked with modern digital transformation projects. Oracle faces the following challenges and opportunities as it drives its vision forward:

- Oracle has continued to make inroads into cloud services, but its late entry into cloud services has meant that many start-ups are already using established cloud services from players such as Amazon Web Services and Microsoft Azure. Nevertheless, Oracle's large base of enterprise customers are in the early stages of mobilizing to the cloud, and the opportunity remains for Oracle to capture much of that transition to its cloud services. Additionally, Oracle's new Application Container Cloud makes it possible for Oracle to attract new workloads where Docker container technology has become increasingly popular.
- Oracle has moved to build a variety of capabilities in Oracle Developer Cloud Service, but the company has to move fast to add new capabilities to take the service to other popular programming languages and frameworks. Oracle is also working to bring application performance management and log processing capabilities and to provide support for bringing external container workloads.

Conclusion

New approaches in software development are taking hold, leveraging the confluence of a number of great practices, tools, and architectures. Microservices and general cloud-native development represent the latest incarnation of this collection of new practices and architectures. Oracle has begun its journey to transform itself to a full-service cloud provider, and on its way, it is providing important new services for customers that are engaged in their own digital transformation journeys. Oracle's new cloud-based developer services are important new services being offered today to Oracle customers and new companies looking for a cloud-based approach to software development.

Global Headquarters: 5 Speen Street Framingham, MA 01701 USA P.508.872.8200 F.508.935.4015 www.idc.com