PERSPECTIVE

Microservices – A New Application Paradigm

Abstract

Microservices Architecture is introducing the concept of developing functionality as a number of small self-contained services. This paper intends to clarify the implications of adopting this approach, appropriate usage, and how microservices can change an organization's approach to integration and delivering functionality through the digital channel.

Infosys
be more

New Integration Model or New Application Paradigm

Microservices is a software architecture and delivery paradigm that delivers functionality by composing a number of small runtime services. Although perceived as a new paradigm, the concept can be traced back to the early days of the Unix Operating System which was developed as a collection of small, simple, and discrete functions.

With microservices, software functionality is delivered as discrete, lightweight services with a well-defined API, which can then be orchestrated to provide a more complete set of business functionality or simply exposed directly to a mobile or web / browser UI. A microservice should be as focused as possible, yet still add value instead of just passing information. This is in contrast to the traditional approach for software application delivery, which is to build, integrate, and test applications as a single integrated monolithic deployment – even if the application provides a set of service interfaces. The microservices approach facilitates faster delivery of smaller incremental changes to an application, at the cost of increased integration and deployment complexity.

In recent years, there has been a significant effort to improve the agility and time-to-market, for changes to the digital UI layer; and at the same time, the underlying core functionality has changed at a much slower speed. With Service Oriented Architecture (SOA), supporting technologies like API Management and ESB platforms can be used to decouple the layers and support the different delivery velocities. Microservices is a mechanism to try and push some of this digital agility back into the core functionality, and align well with agile delivery and the DevOps operating model.

These days the term Microservices tends to refer to developing functionality as a collection of small services, each running in its own process and accessed via a lightweight interface, such as an HTTP RESTful API
- Fowler

Microservices – Part of the Digital Economy

The Microservices Architecture is well aligned to the move towards cloud-based SaaS offerings and the API economy. In some ways, it can also be traced back to the early days of SOA with the focus on service discovery, and anonymous service usage based on UDDI. What has changed is the proliferation of RESTful APIs and the popularity of hybrid service integration of cloud SaaS services plus the availability of virtualised platforms and containers, together with the advanced tools to manage them.

A Microservices Architecture can provide key technology business opportunities as a service provider, service consumer, or increasingly as both a provider and consumer.

- Microservices support consumer enablement, transparency, and choice
- Microservices can extend the utility and reach of your business
- Microservices can provide technology agility and thus improve the business' time-to-market

For a service provider, microservices should be considered as digital business assets and treated accordingly. They need to have a business owner and a business strategy / model associated with them. If the focus is on revenue generation, then services need to be promoted and the business needs to monitor usage, and simultaneously engage with the community to ensure the services are relevant, in demand, and competitive. If competitive advantage is the focus, then the services need to be relevant, responsive to change, and have a planned life cycle including obsolescence planning.

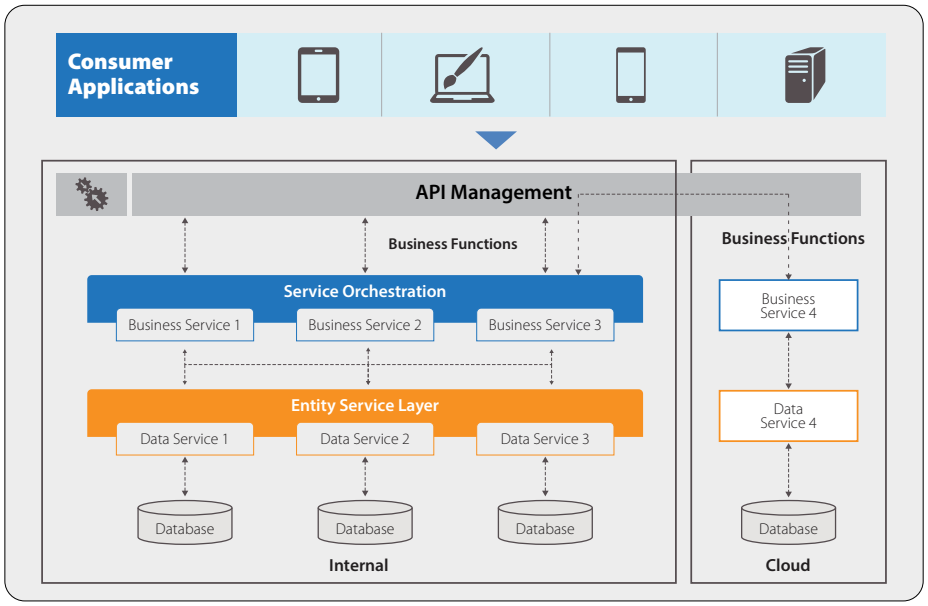
As a service consumer you need to be continually checking the market for new services, providers, and opportunities aligned with your business strategy.

Organizations are increasingly looking at microservices as a valuable supplement to internal application services being delivered through the digital channel. Instead of extending internal capability, external microservices are being aggregated or enriched, and then provided as value-added microservices to the outside world.

Although a Microservices Architecture might be initially focused on supplementing internal functionality for users, it also provides opportunities to take an existing service, value-add or supplement it, and then deliver the enhanced service out to a different consumer audience. It can be as lightweight as brokering and rebranding for marketing advantage, or aggregating with other services to provide a more functionally rich microservice offering.

How do Microservices Impact Application Strategy?

Historically, organizations would look to monolithic application suites to solve business problems such as ERP, HCM, or CRM. Evaluating functionality was based on an overall assessment of a small number of platforms against key business requirements. However, with cloud / SaaS microservice providers, this selection has become much more complex, but at the same time, much more flexible. Organizations are no longer constrained by vendor upgrade schedules, but are instead able to quickly bring new functionality to the online channel whenever new or enhanced microservices become available. This is particularly valuable for mobile and browser digital-applications as they are able to quickly make use of new distributed microservices.



So the new approach in assessing functionality is to look for stability around a core set of services for a minimum viable product (MVP) and then be opportunistic with enhancements as new services become available for use. When working with microservice providers who are effective at engaging the consumer community, there are options to influence functionality direction, to better support new business directions. Thus assessment criteria will now need to consider additional aspects such as:

- Community engagement
- Alternative service options
- Supplier longevity and reliability

This effectively moves emphasis away from single source providers towards a best-of-breed service provider approach. However, the granularity is much smaller, emphasising the need to leverage lightweight engagement approaches rather than heavyweight contractual approaches for usage of services outside of core, long-term dependencies.

This lightweight approach drives organizations to replace traditional application / asset owner models with a more lightweight service owner model within the business – both as a service consumer (owning the relationship) and as a service provider (having the marketing and management responsibility). The approach also needs to take into account the MVP functionality aspect to ensure those core services are recognized as core and are sustainable in the future. These MVP services will need to have tighter governance around change.

The other key consideration for selecting microservices is data governance. Privacy legislation and other compliance needs can restrict storage of data off-premises or outside country boundaries. This is another key assessment criterion when assessing microservice providers

Microservice Architecture focuses on building scalable, distributed applications that support agile deployment – both on-premises and to the cloud. The benefits and complexities of Microservices Architecture need to be carefully considered before taking this approach to delivering application services
– Gartner - Olliffe

How do Microservices Change Integration?

The increased prevalence of cloud-based SaaS providers in the enterprise-business-functionality mix has broken down the monolithic application approach and increased the need to integrate multiple services, applications, and providers, to support the enterprise. A potential impact of microservices is to dramatically increase independent service numbers, with the corresponding increase in integration complexity.

As microservices segment functionality into smaller and simpler services – in order to achieve more complex, integrated functionality – they can give rise to a significant increase in the demands on an organization's ESB / orchestration platforms. There is also an increased likelihood that these microservices will be hosted and provided through cloud / SaaS providers, which drives an enterprise towards a hybrid integration approach to better support external service aggregation, and to integrate external microservices into the enterprise. This results in an increased emphasis on hybrid mediation and orchestration platforms as a key enablement capability within an enterprise.

As an enterprise acting as a service consumer, mediation and orchestration are crucial to effectively integrating services internally; whereas from the point of view of a microservice provider, API Management platforms are crucial to managing service access. An API-Management-type platform provides the security and service management capabilities to support quality delivery of RESTful microservices, and also enables valuable social engagement and collaboration functionality to support engaging the consumer community. These tools and platforms not only support managing the life cycle of services, but more importantly provide opportunities to more effectively monetize the provided microservices.

Operational Impacts of a Microservices Architecture

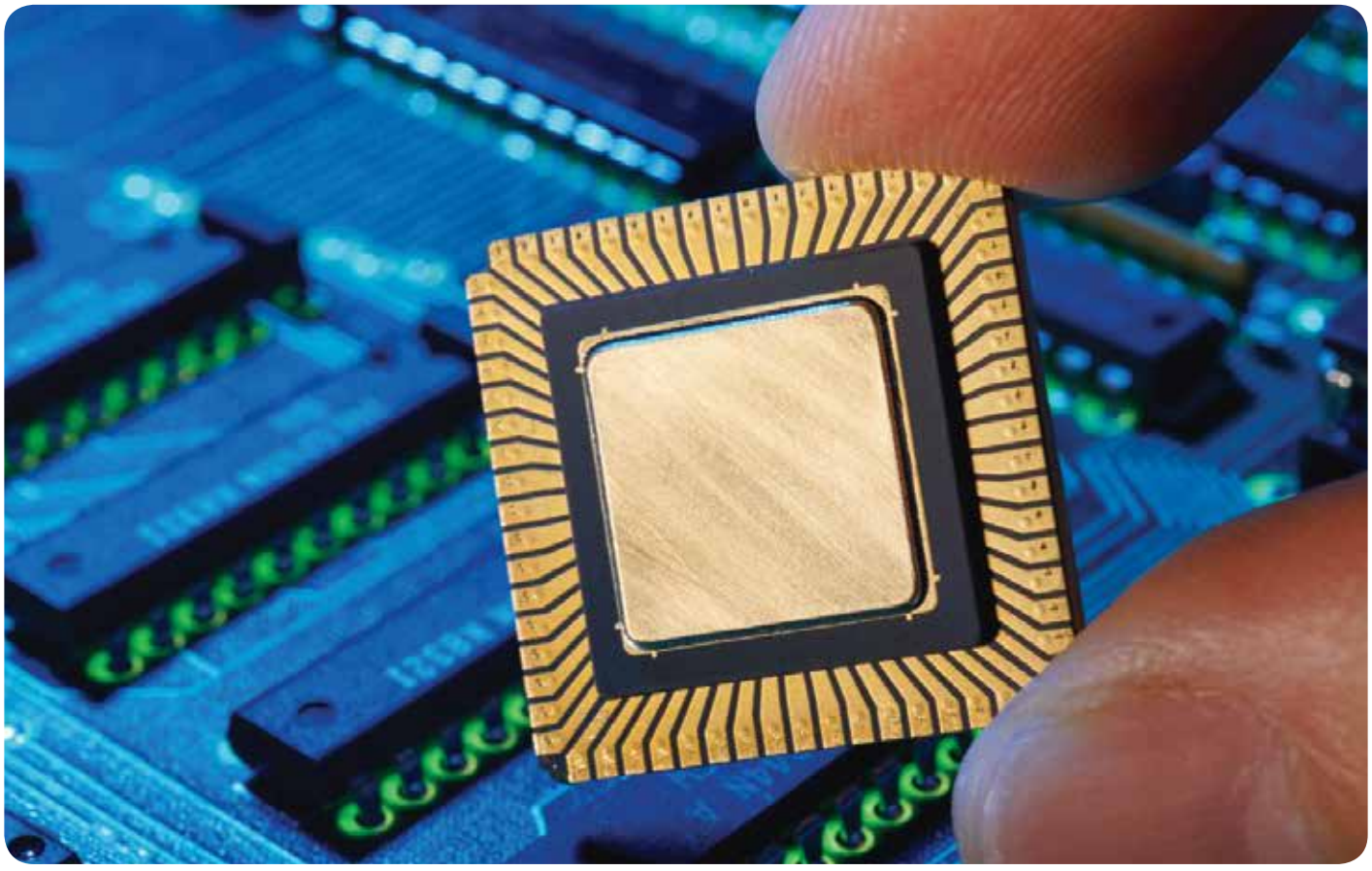
Managing microservices is heavily dependent on having the right tools and platforms in place. There needs to be effective support for aspects such as container hot deployment, and tools to support flexible runtime deployment and automated monitoring.

As noted above, the use of a Microservices Architecture can increase the complexity of providing enterprise functionality. This drives a heavy dependency on integration and operational management technologies to handle the complexity. At the same time, a Microservices Architecture provides an opportunity to simplify digital service delivery, where there is good alignment between the available microservices and the user functionality provided.

In general, an enterprise Microservices Architecture approach requires support from modern-day, advanced tools and monitoring capabilities to handle the complexity. This results in:

- Requirements for effective distributed management tools and dashboards
- The need to design for resilience to ensure usability when service availability is beyond the control of the enterprise
- The need to move towards a more agile, continuous change management and delivery mode to effectively manage service life cycle changes and replacement of microservices
- A shift towards a more lightweight release process to align with incremental service change versus major application replacement / upgrade. At the same time, major deployments can be more complex to ensure all the required microservices are deployed correctly
- A need for comprehensive service monitoring to ensure availability of microservices, to replace what would have previously been vendor-specific application monitoring tools. Also, a single microservice failure will not be as obvious as an application failure
- Changes to operational support as L2 / L3 support will need to be more fine-grained at the individual service level
- An accelerated transition from application asset ownership towards service ownership

It also reinforces agile delivery and testing best practices around automation and continuous delivery.



Microservices Best Practice

Microservices and their APIs must be managed as business products, from conception to retirement

- Need to align with business strategy
- Need to provide a revenue stream or competitive advantage
- Need to be marketed and managed as business assets

Control service access based on business requirements, not technology constraints

- Once microservices are in the wild it is impossible to control consumers
- Consumers will often identify ways to use a service that is radically different from how it was originally envisioned
- Focus on supporting industry-standard identity management protocols, rather than custom mechanisms

Design for service unavailability

- Need to provide resilient behavior over microservices by designing for flexibility and temporary outages

Manage SLAs

- Agree to well-defined service SLAs and proactively manage the performance

Service discovery and definition

- Ensure microservices are well-defined and documented, in an online repository or catalog with simple consumer access

Microservice granularity

- Services granularity should be aligned with the RESTful API model (CRUD)
- Services should provide a useful business service
- Data services should not be exposed directly unless they provide a business function

Life cycle management

- Services specification must be designed to support versioning and eventual depredation

Hybrid integration

- Aggregate, enrich, or orchestrate microservices at the appropriate place in the Architecture (business functionality). This is not within the mobile or browser app

Data Management

- The usual approach is for each microservice to "own" its data (Gartner - Thomas) – which matches with the CRUD-style service interface. This approach goes back to traditional Object-oriented principles, which have been superseded by SOA approaches. This has a tendency to drive microservices APIs towards being low-level data services based upon data schema, but this approach should be resisted to ensure alignment with delivering business value rather than alignment with an underlying data schema
- Data should be explicitly replicated where needed, using a master data management paradigm, rather than relying on implicit propagation which may fail due to service provider changes
- Look for opportunities to use in-memory data grid technology for low latency and scaling across microservices. This aligns well with the model of a data microservice, with a small defined set of data rather than focusing on a large integrated data model

Microservices Architecture – Is it Right for You?

All modern day digital architectures should be service-enabled through HTTP / SOAP web services, RESTful APIs, JMS queues, etc. Unless an organization runs on a single application platform, this is essential to support effective integration across the enterprise and deliver an integrated user experience. A Microservices Architecture is a very specific variation of a sservice-enabled architecture which is more focused on agility. It can provide significant value in a number of areas such as:

- For a rapidly changing / growing business or business group where time-to-market is key and the core business is more dynamic and changing
- Test the market situations where functionality needs to be rolled out in a low-cost manner focused on validating the demand
- Mobile application development or mash-up browser application development based on RESTful APIs and not requiring deep integration into existing core systems
- For supplementing the functionality of an out-of-the-box application or SaaS service to minimize customization and thus simplify future application upgrades

In general, there is real value to an organization in having an API-enabled, service-oriented architecture. Microservices provide an added advantage by also supporting service agility, at the potential cost of increased integration complexity. This can work well for smaller installations but needs careful evaluation around the complexity and risk implications for larger installations.

Infosys Experience with Microservices

Digital Alignment

Assisting a leading Australian online trading service provider to move from a centralized application platform over a number of SaaS service providers, to a more loosely coupled architecture. This was based on directly exposing the underlying microservices APIs, where appropriate, for direct consumption by mobile and browser apps; and where more complex behavior is required, vendor APIs are aggregated via an internal integration platform to provide more complex microservices for consumption.

ESB-based Microservice Architecture

The channels division within a major international bank has embarked on a journey to transform their existing technology stack to a next generation landscape to help deliver a unified digital experience to all of its online customers. Infosys provided an innovative channel services layer architecture based on Fuse ESB, which has built-in support for Apache Camel for orchestration, and Apache CXF for hosting web services. The solution is leveraging the loose coupling of OSGi bundles and the orchestration capabilities of Apache Camel framework to build microservices that allow independent deployment. Fabric, the clustering mechanism of Fuse, will ensure that these services can then be horizontally scaled on demand.

References

Fowler, J. L. *Microservices.* Retrieved from Martin Fowler: http://martinfowler.com/articles/microservices.html

Gartner - Olliffe, G. *Assessing Microservice Architecture for Scalable Application Delivery.* Retrieved from Gartner: https://www.gartner.com/doc/2974417/assessing-microservice-architecture-scalable-application

Gartner - Thomas, A. *Application Architecture for Digital Business.* Retrieved from Gartner Webinars: http://my.gartner.com/portal/server.pt?open=512&objID=202&mode=2&PageID=5553&ref=webinar-rss&resId=2866521&srcId=1-3134301641

About the Author

Peter Jarman

Principal Technology Architect , Digital and Integration Services, Infosys

Peter Jarman is a Principal Technology Architect for Digital and Integration Services at Infosys. He has over 30 years' experience in IT, covering a wide range of business domains. He has significant experience across EAI, BPM, SOA, Digital / Online, Enterprise Architecture, and Solution Architecture, and has published a variety of technical papers on these topics.

For more information, contact askus@infosys.com

Infosys be more