



Changing how we collect
store, manage, transform
and analyze clinical data,
one microservice at a time

— — — —



Mark Bynens, Janssen R&D

Paul Fioole, Janssen R&D

Disclaimer

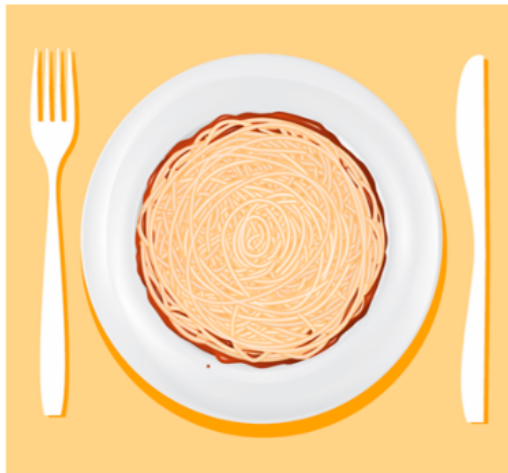


The opinions expressed are those of the authors and do not necessarily represent the opinions of PHUSE, members' respective companies or organizations, or the products' respective companies mentioned in this paper.

It is the reader's or audience's responsibility to determine where microservices would best suit their need.

Additionally, the authors do not endorse any specific commercial products or services.

Software Architecture Evolution



With monolithic, tightly coupled applications, all changes must be pushed at once, making continuous deployment impossible.



Traditional SOA allows you to make changes to individual pieces. But each piece must be carefully altered to fit into the overall design.



With a microservices architecture, developers create, maintain and improve new services independently, linking info through a shared data API.

<http://www.kanbansolutions.com/blog/microservices-architecture-friend-or-foe/>

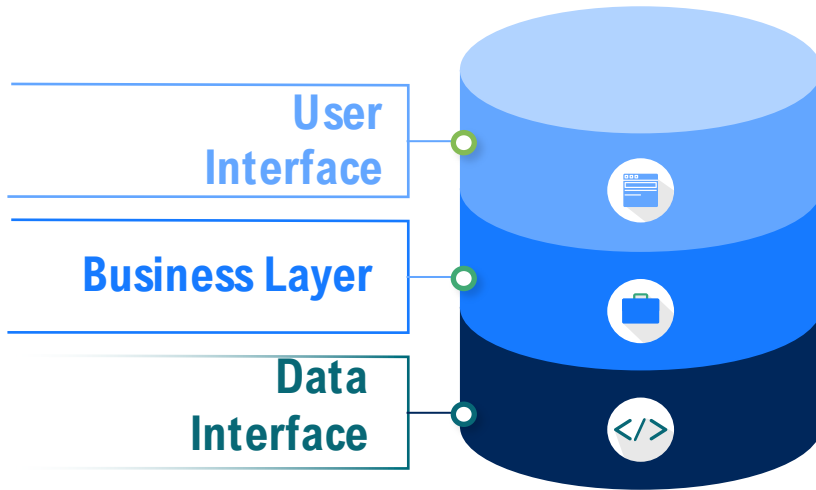
What are microservices?



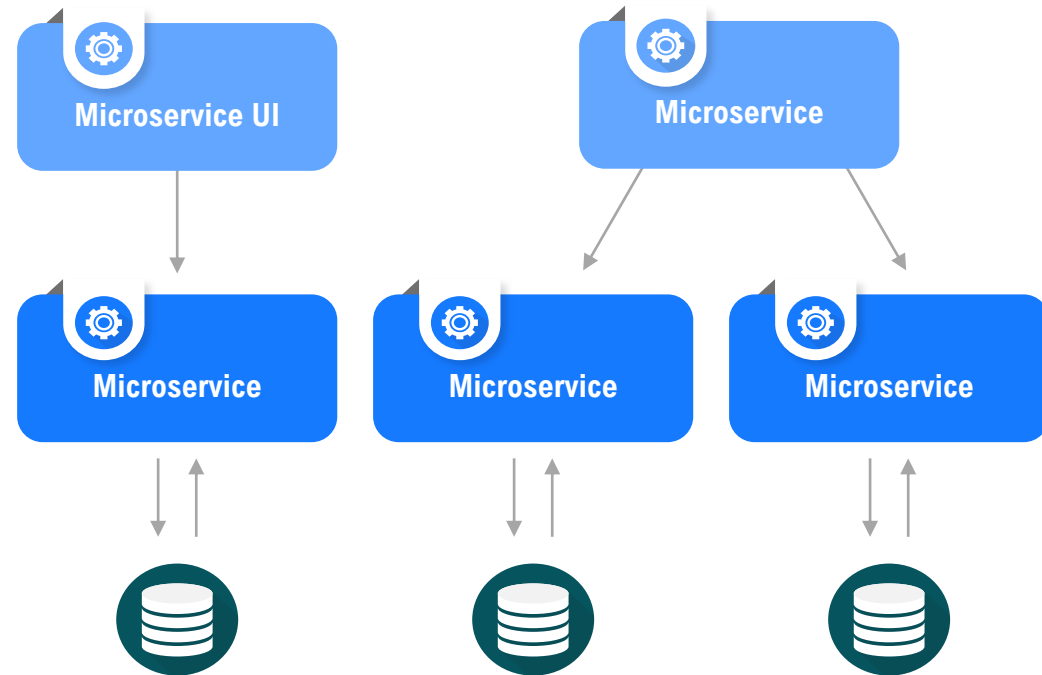
Microservices are an architectural approach to software development that structures an application as a collection of small autonomous services, modeled around a business domain.

Loosely coupled service-oriented architecture with bounded context - Adrian Cockcroft

MONOLITHIC ARCHITECTURE



MICROSERVICES ARCHITECTURE



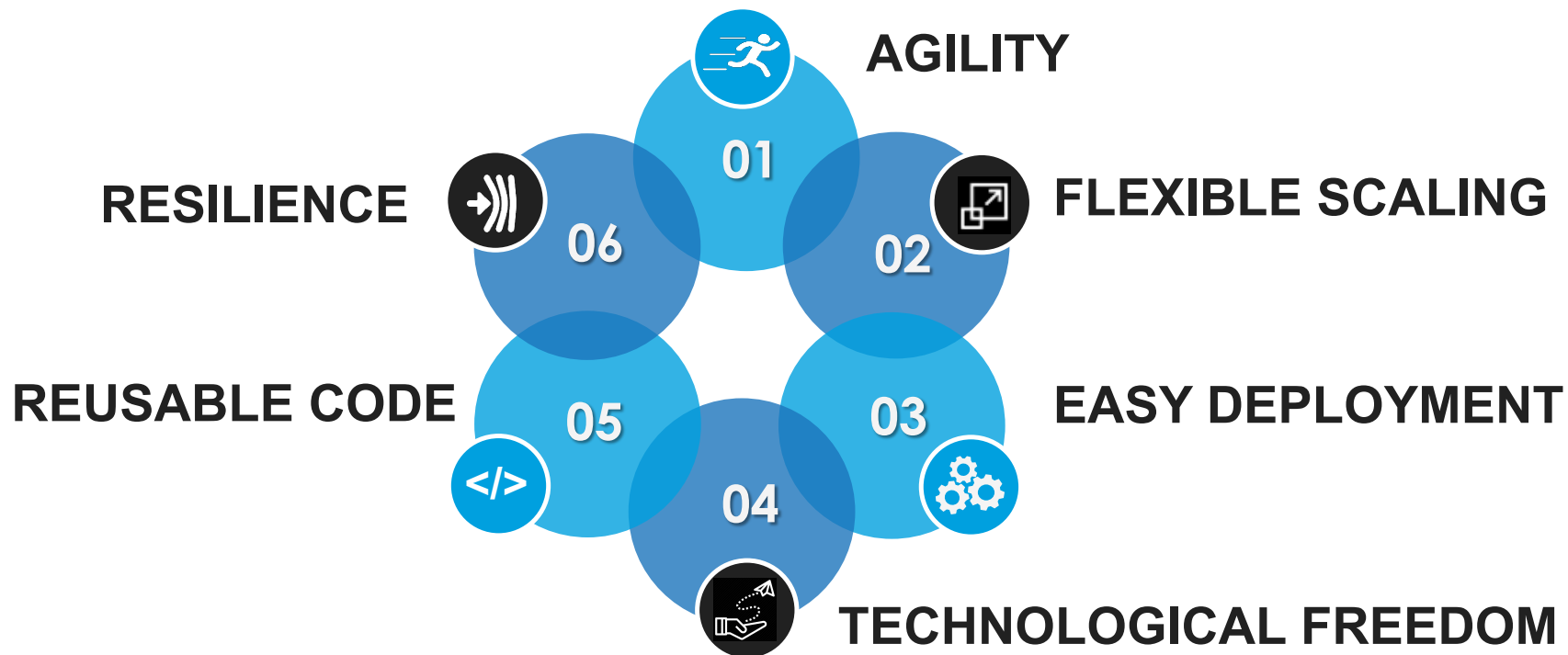
There are multiple characteristics of microservices. Some of them are

Characteristics



- SPECIALIZED
- AUTONOMOUS
- LOOSELY COUPLED
- DECENTRALIZED
- BUILD IT, RUN IT
- POLYGLOT
- BLACK BOX

Benefits of Microservices



Benefits of Downsides



- ✖ Complexity of a distributed system to manage and integrate the microservices
- ✖ Because the components are distributed, end-to-end testing can become complicated and tedious
- ✖ Developers must put additional effort and attention/rigor into implementing microservices
- ✖ Microservices require experienced developers with a very high level of expertise

When to use and when not to use microservices



TO USE

- When you want your monolithic application to accommodate scalability, agility, manageability and delivery speed
- When you have to rewrite legacy applications in today's programming languages or tech stacks to keep up with modern-day business requirements and solutions
- When you have standalone business applications or modules that have to be reused across diverse channels. Some good examples would be login services, search options, authentication facilities, ...

When to use and when not to use microservices



NOT TO USE

- Complexity, more than anything else, is the problem for which microservices are a solution
- Using microservices can prove to offer contrary consequences if you don't have separate teams to handle microservices independently
- Some applications by their nature require tight integration between individual components and services.

Use case in Pharma

Historical ecosystem in clinical environments

- Individual automation solutions – frequently departmentally
- Complex interactions and interfaces

Commercial offerings

- One-stop solutions, frequently monolith approach
- Leverage supplier for software and interfaces
- Requires process to be aligned with supplier system



Microservices approach

- Allows for gradual implementation
- Leverage existing components and (cloud) services
- Adapt to business processes
- Technical flexibility in a changing environment
- Less risk of lock-in to a single platform

Use case – Legacy vs Microservices



Our Legacy SCE

- Single SCE application server - fixed capacity – stand alone SAS compute servers
- Op premises deployment, distributed over 2 internal datacenters to cope with regional access
- Internal complexity of the web app had grown over the years, increasingly difficult to make changes or upgrade to a new software releases
- Software was written by one (inhouse) development team, in a very integrated way

Use case – Legacy vs Microservices



‘ μ S’-based SCE

- Flexible capacity through grid compute models and container based microservices deployment
- Cloud based environment, fully web based, costing schemes are adapted to capacity needed
- Software is written in separate microservices, loosely connected via API's, allows for more effective lifecycles
- Functionality has been developed as separate modules, where new modules can be added for new functionality, these can be written in different languages or by different teams. Where applicable, cloud services are leveraged



Email Addresses

Mark Bynens: MBYNENS@its.jnj.com

Paul Fioole: PFIOOLE@its.jnj.com



Thank You



PHARMACEUTICAL COMPANIES OF

Johnson & Johnson



**Creating a future where
disease is a thing of the past**