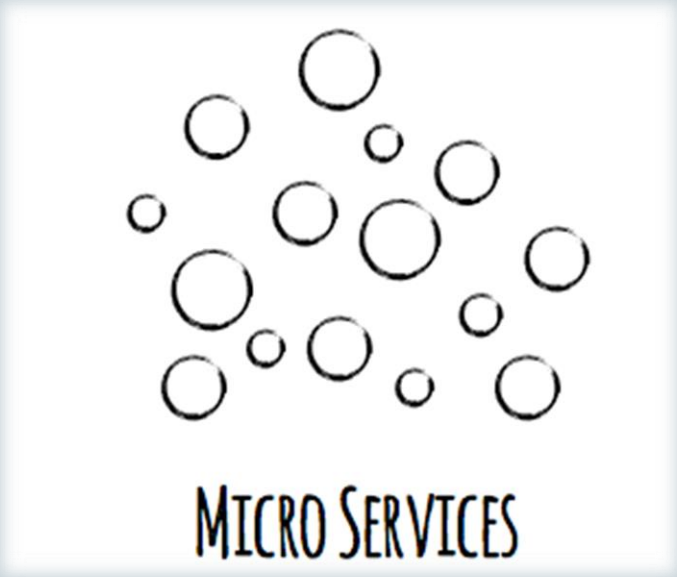
Micro Services in the cloud



MEGATRIS COMP. LLC

Web services

It is a software function provided at a network address over the Web with the service always on as in the concept of utility computing.

The W3C defines a Web service generally as:

a software system designed to support interoperable machine-to-machine (M2M) interaction over a network.

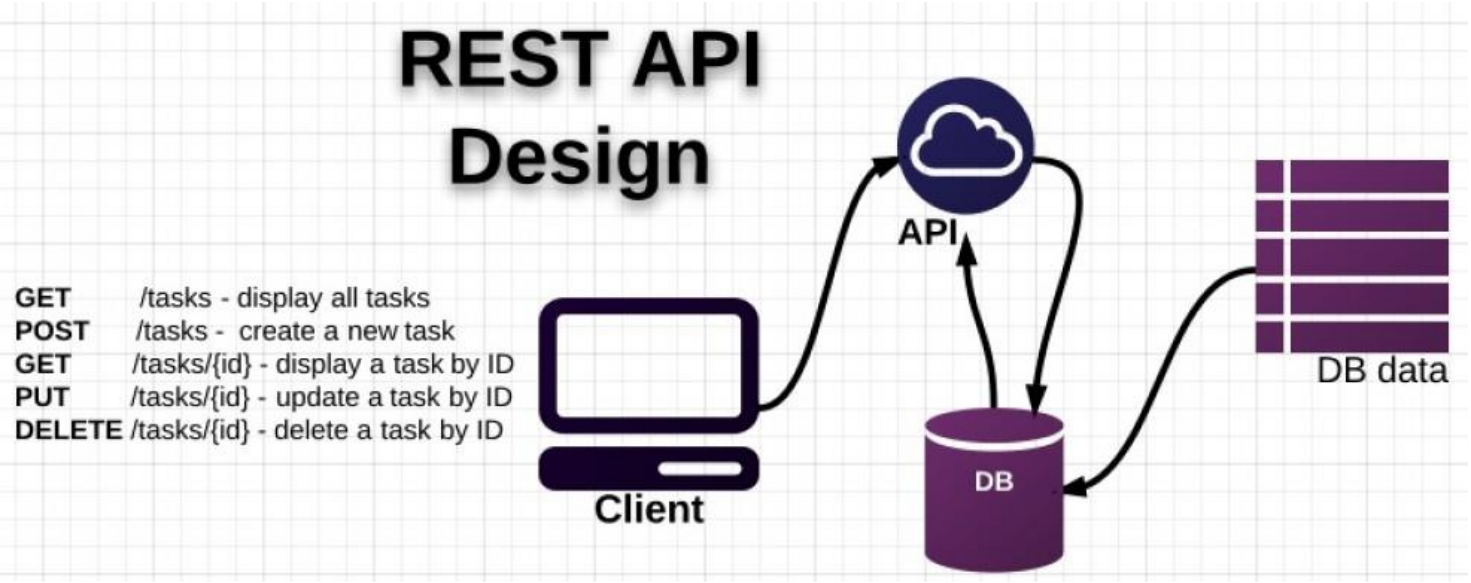
API

A server-side web API is a programmatic interface to a defined request-response message system, typically expressed in JSON or XML, which is exposed via the web.

While "web API" in this context is sometimes considered a synonym for web service, Web 2.0 web applications have moved away from SOAP-based web services towards more cohesive collections of **RESTful** web resources. These **RESTful** web APIs are accessible via standard HTTP methods by a variety of HTTP clients including **browsers and mobile devices.**

REST

REST stands for Representational State Transfer.
It relies on a stateless, client-server, cacheable communications protocol -- and in virtually all cases, the HTTP protocol is used. **REST** is an architecture style for designing networked applications.



Monolithic Application

Networked or offline applications can easily be designed using a monolithic approach.

A monolithic application built as a single unit. Enterprise Applications are often built in three main parts:

- a client-side user interface,
- a database
- a server-side application.

The server-side application will handle HTTP requests, execute domain logic, retrieve and update data from the database, and select and populate HTML views to be sent to the browser. This server-side application is a *monolith* a single logical executable. **Any changes to the system involve building and deploying a new version of the server-side application.**

Microservices

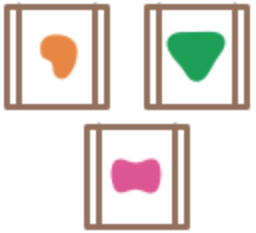
Instead of using a monolithic approach, the microservice architectural style is an approach to develop a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

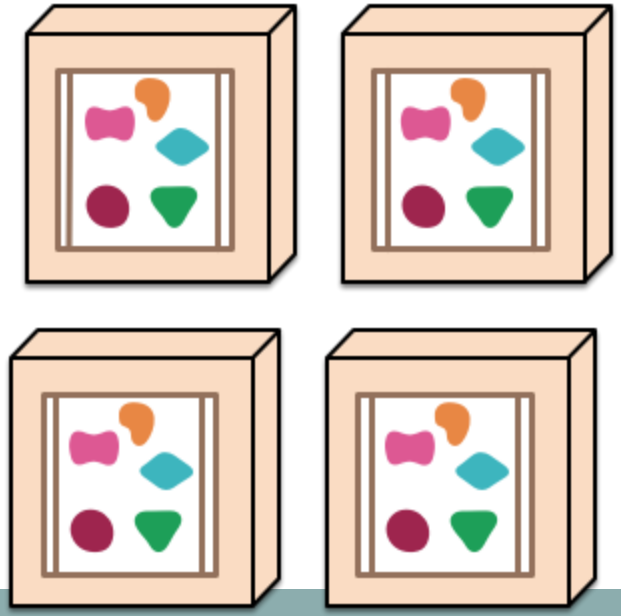
Monolithic VS Microservices

A monolithic application puts all its functionality into a single process...

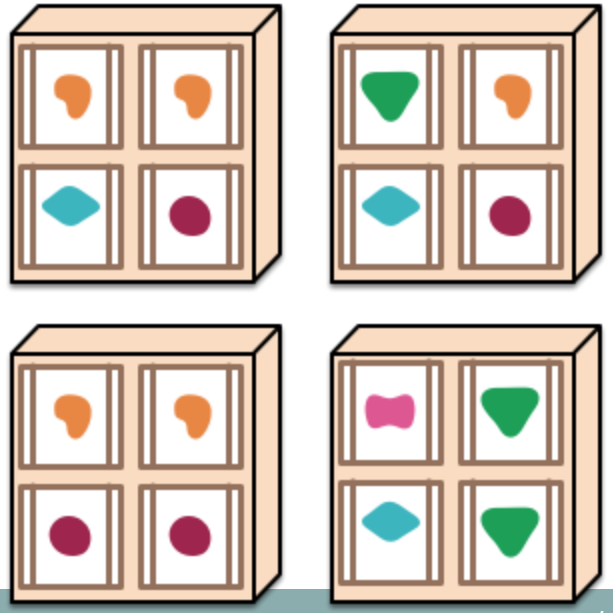
A microservices architecture puts each element of functionality into a separate service...



... and scales by replicating the monolith on multiple servers



... and scales by distributing these services across servers, replicating as needed.



Microservices win!

These frustrations have led to the microservice architectural style: building applications as suites of services**. As well as the fact that services are independently deployable and scalable**, each service also provides a firm module boundary, even allowing for different services to be written in different programming languages. **They can also be managed by different teams.**

The four pillars

1. The services are small - fine-grained to perform a single function.
2. The organization culture should embrace automation of deployment and testing. This eases the burden on management and operations.
3. The culture and design principles should embrace failure and faults, similar to anti-fragile systems.
4. The services are elastic, resilient, composable, minimal, and complete.

Do one thing and do it well.

IoT & Microservices

IoT (Internet of things) is expanding. More and more devices are now connected, big companies like Samsung, Google and LG are working on the home consumer sector to create a simple way to **connect everything to internet** (IoE – internet of everything).

Every device becomes connected and intelligent thanks to the cloud.
We strongly think that a Microservices architecture based on **cloud services** can monitor, analyze data and send- receive commands to/from every object in the IoT cloud.

The object it's not only a thing, but also an agent, that using his mobile devices has the ability to use microservices.

It's clear that we need a simple way to manage so different sources of data and **Microservices are the key** to do that.

Achitecture based on Microservices

When talking about components we run into the difficult definition of what makes a component. Our definition is that a **component** is a unit of software that is independently replaceable and upgradeable.

The microservices architecture primary way of componentizing their own software is by breaking down into services.

Services

Services are out-of-process components who communicate with a mechanism such as a web service request, or remote procedure call.

One main reason for using services as components is that services are independently deployable. If you have an application that consists of a multiple libraries in a single process, a change to any single component results in having to redeploy the entire application. But if that application is decomposed into **multiple services,** you can expect many single service changes to only require that service to be redeployed. The aim of a good microservices architecture is to minimize these through cohesive service boundaries and evolution mechanisms in the service contracts.

Contextual Microservices architecture for the IoT

A possible solution to easily control the IoT systems is to create an intelligent platform using a microservices architecture.

Each service has one and simple behavior and it's called when a specific event occours in the system.

Everything is event driven and the flow from the start to the end is influenced by the **context**.

Context

*"**Context** is any information that can be used to characterize the situation of an entity.*

An entity is a person, place or an internet object that is considered relevant to the interaction between a user and an application, including the user and application themselves."

How to build this architecture?

The most difficult aspect of a contextual microservices architecture is to build a system that has the flexibility, intelligence and complexity capable of manage the net of microservices.

We call such layer **System Bus.**

System Bus

THE GLUE TO CONNECT DIFFERENT TIERS

System Bus

The **system bus** was born as a single computer **bus** that connects the major components of a computer **system**. The technique was developed to reduce costs and improve modularity.

We applied this idea to the Microservices architecture to glue services with complex processes, mobiles, things, etc.

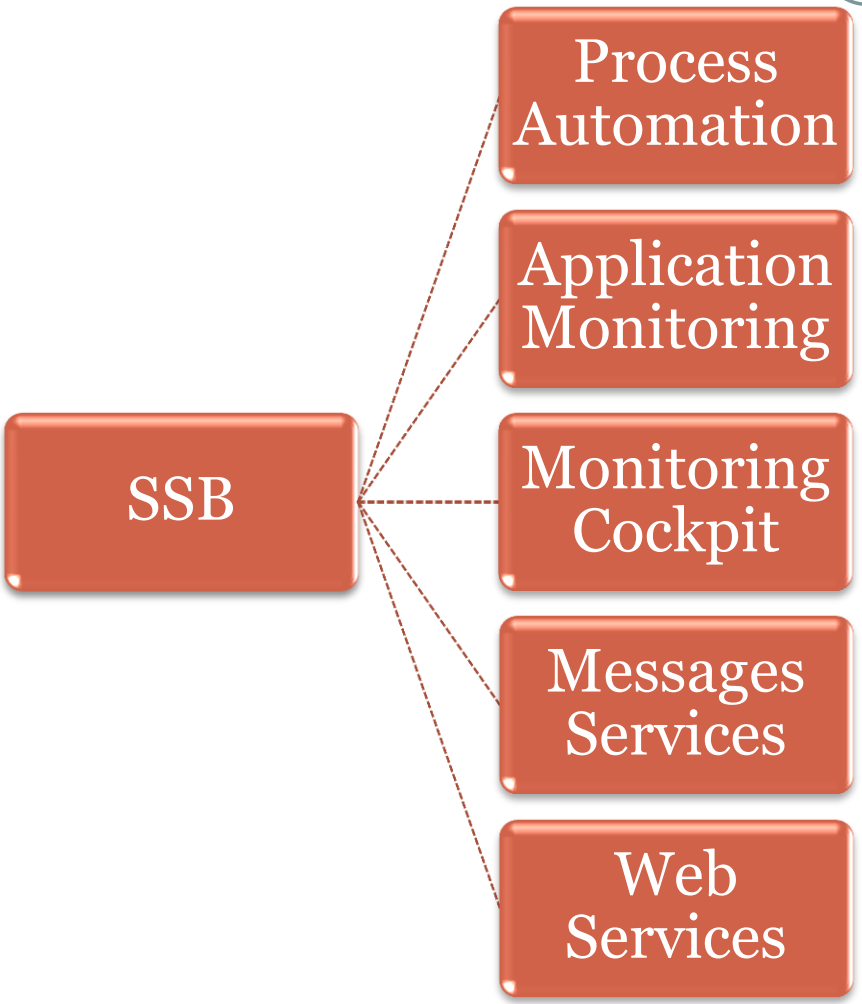
Our System Bus

We want to give a real life example introducing **SSB, our event driven enterprise cloud system bus for the IoT.**

SSB permits Interaction Services using Web Services (WS) and real time simulations.

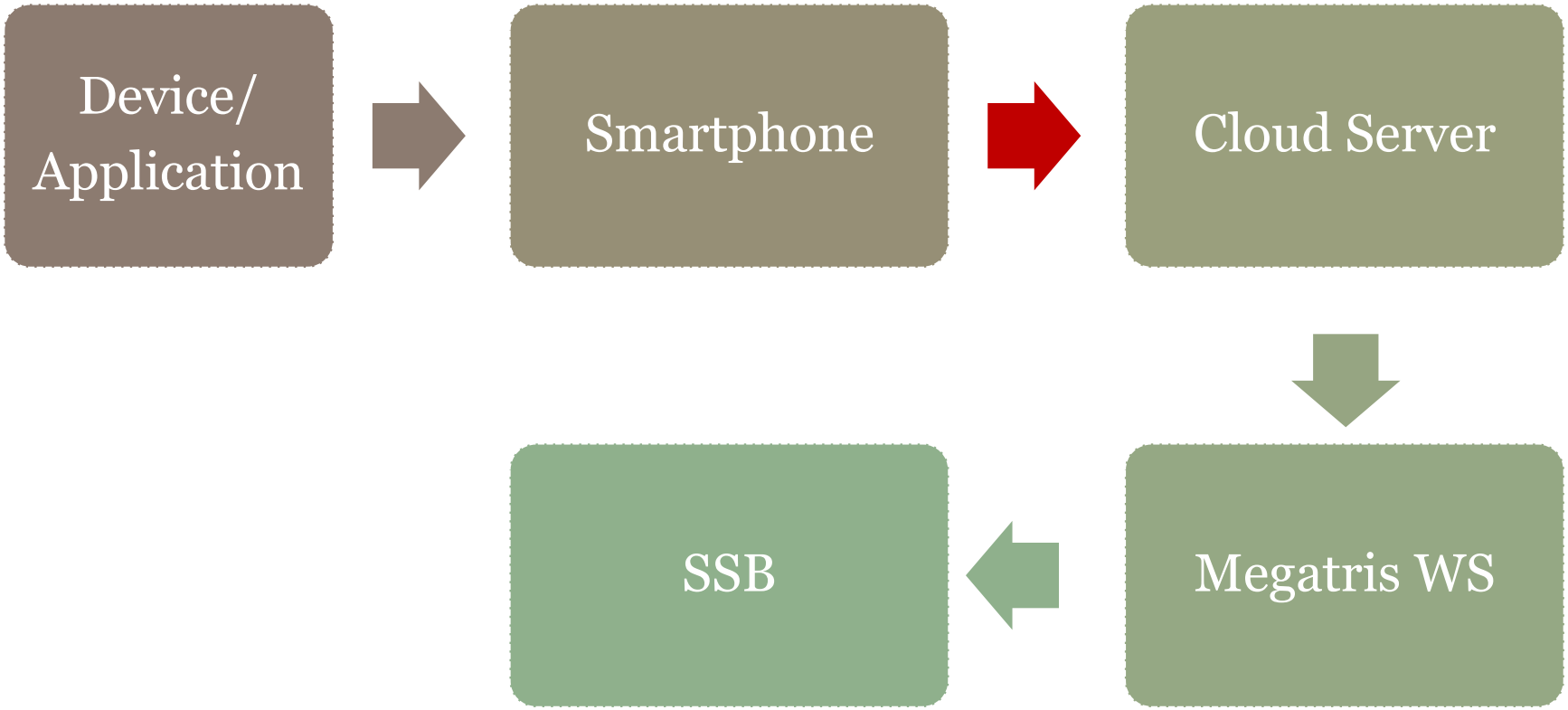
SSB platform is a model based and event-driven application with directive constraints and actions.

The architecture



- SCC (simulation and control manager)
- Communication server (manages communication using email, sms, twitter, push, …)
- IMCP (a rule base decision maker)
- Extractor (prepare data using specific rules)
- A suite of web services

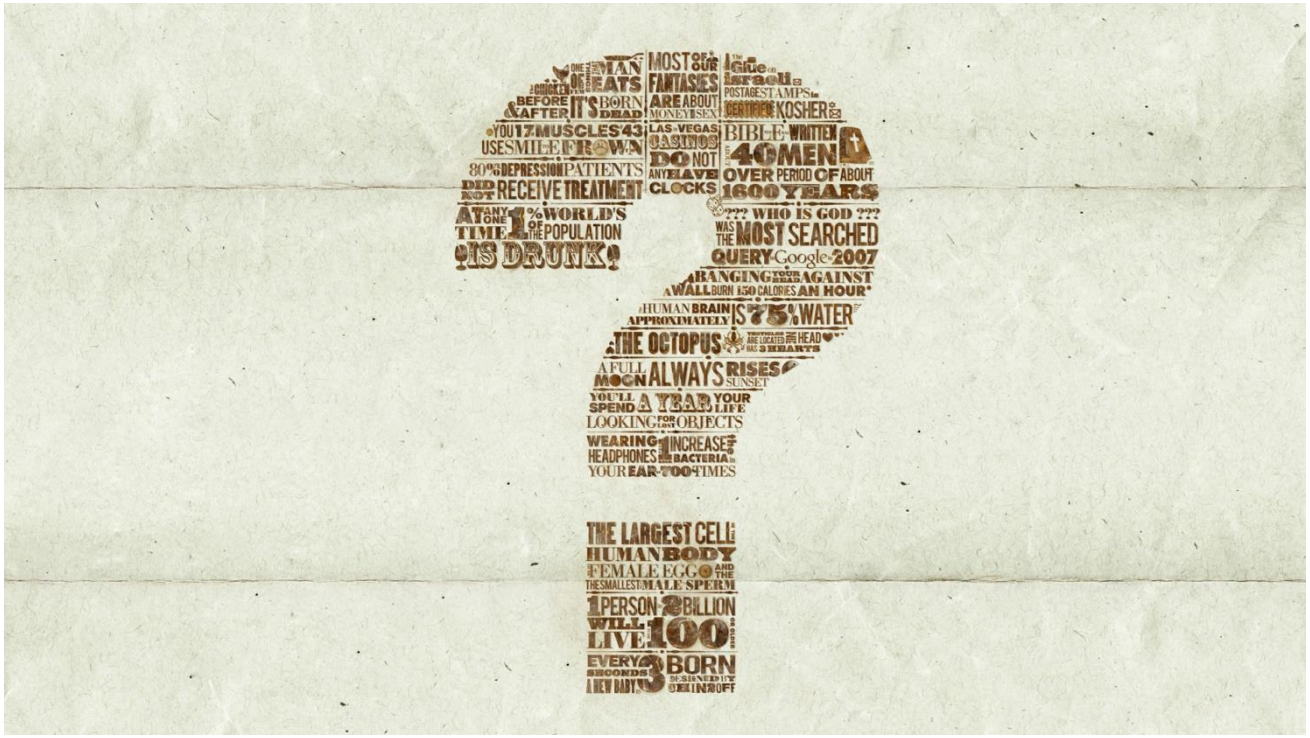
Architecture Flow



Megatris Comp. LLC

We create cloud services and mobile apps to make people life easier.
Our mobile apps are integrated with Megatris Cloud to sell services and goods.

www.megatris.com
1250 Oakmead Pkwy, Sunnyvale, CA 94085, USA


MAN EATS
BEFORE & AFTER IT'S BORN DEAD
MOST OF OUR FANTASIES ARE ABOUT MONEY/SEX
LAS VEGAS CASINOS DO NOT ANY SLAVE CLOCKS
BIBLE WRITTEN OVER 40 MEN OVER PERIOD OF ABOUT 1600 YEARS
80% DEPRESSION PATIENTS DID NOT RECEIVE TREATMENT
AT ANY ONE TIME 1% OF THE WORLD'S POPULATION IS DRUNK!
??? WHO IS GOD ???
WAS THE MOST SEARCHED QUERY Google 2007
BANGING YOUR HEAD AGAINST A WALL BURN 150 CALORIES AN HOUR*
HUMAN BRAIN APPROXIMATELY IS 75% WATER
THE OCTOPUS HEAD HAS 3 HEART'S
A FULL MOON ALWAYS RISES SUNSET
YOU'LL SPEND A YEAR YOUR LIFE LOOKING FOR OBJECTS
WEARING HEADPHONES INCREASE BACTERIA YOUR EAR 700 TIMES
THE LARGEST CELL HUMAN BODY
FEMALE EGG AND THE SMALLEST MALE SPERM
1 PERSON 2 BILLION WILL LIVE 100
EVERY 3 SECONDS A NEW BABY BORN

QUESTIONS?

EXTRA SLIDES

Event Driven

An event can be defined as "a significant change in state”. For example, when a consumer purchases a car, the car's state changes from "for sale" to "sold". **Event-driven** programming is a programming paradigm in which the flow of the program is determined by **events** of all kind.

So, an event in a Micro Services architecture becomes the trigger that can activate a flow between a small or huge amount of layers.

Service-oriented architecture

A **service**:

- Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports)
- Is self-contained
- May be composed of other services
- Is a "black box" to consumers of the service

SOA is based on the concept of a service. Depending on the service design approach taken, each SOA service is designed to perform one or more activities by implementing one or more service operations.

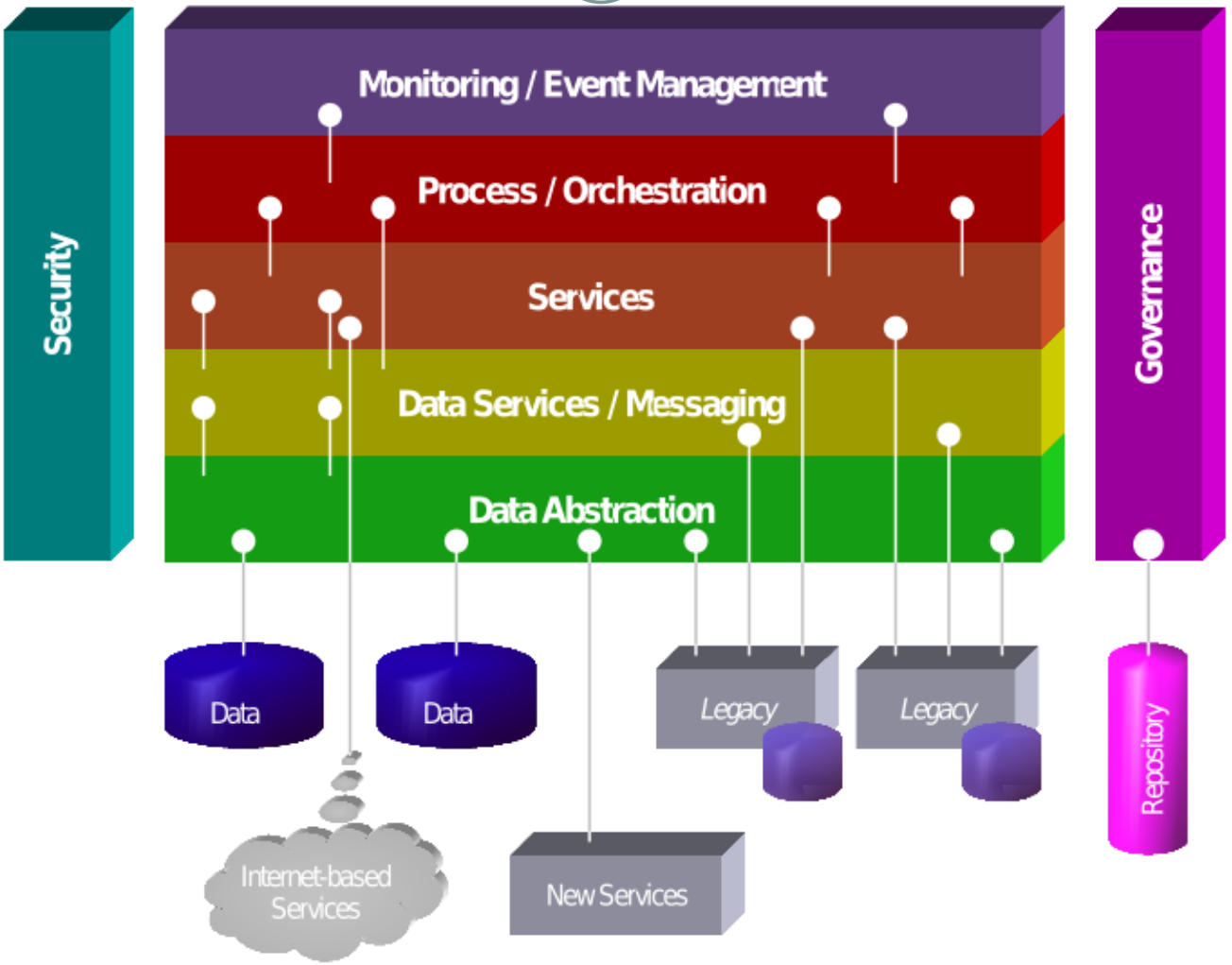
Services are unassociated, loosely coupled units of functionality that are self-contained.

Service-oriented architecture

Programmers have made extensive use of XML in SOA to structure data that they wrap in a nearly exhaustive description-container. Analogously, the Web Services Description Language (WSDL) typically describes the services themselves, while SOAP (originally Simple Object Access Protocol) describes the communications protocols.
Each interface brings with it some amount of processing overhead, so there is a performance consideration in choosing the granularity of services.

SOA as an architecture relies on service-orientation as its fundamental design principle. If a service presents a simple interface that abstracts away its underlying complexity, then users can access independent services without knowledge of the service's platform implementation

SOA MODEL


Security
Monitoring / Event Management
Process / Orchestration
Services
Data Services / Messaging
Data Abstraction
Governance
Data
Data
Legacy
Legacy
Repository
Internet-based Services
New Services