

Eng. E Gestão de Serviços (EGS)

► Introdução

Diogo Gomes dgomes@ua.pt

“Software is eating the world”

Marc Andreessen, 2011

- ▶ Every business you can think of has either been digitalized or is in the process of becoming digitalized
- ▶ Large sums of money are being put into this unstoppable process
 - ▶ EU is financing with €7.5 billion in H2030 Digital Europe Programme (<https://digital-strategy.ec.europa.eu/en/activities/digital-programme>)
 - ▶ US Capital investment in 2020 alone: \$126 billion (statista.com)

amazon

R
Revolut

Uber

New industries

It's not just about replacing old industries, software is creating their own new industries

Most importantly users are supported through smart mobile devices:

79% of FB users are mobile only (2020)

60% of individuals have access to the internet (2021 - ITU-T)

2019 Worldwide Download Leaders									
Games		Social/Messaging		Entertainment		Food & Drink		Dating	
1	Garena Free Fire 266.9M	1	WhatsApp 759.4M	1	Netflix 236.6M	1	Uber Eats 96.8M	1	Tinder 78.4M
2	Subway Surfers	2	TikTok	2	YouTube	2	McDonald's	2	Badoo
3	Color Bump 3D	3	Messenger	3	Hotstar	3	DoorDash	3	Tantan
4	Fun Race 3D	4	Instagram	4	Amazon Prime Video	4	Swiggy	4	Happn
5	Run Race 3D	5	Facebook	5	YouTube Kids	5	Starbucks	5	Bumble
6	Homescapes	6	Snaphchat	6	ZEDGE Wallpapers	6	foodpanda	6	MeetMe
7	Tiles Hop	7	Facebook Lite	7	JioTV	7	Rappi	7	Plenty of Fish
8	My Talking Tom 2	8	Pinterest	8	Watermelon Video	8	iFood	8	LOVOO
9	PUBG MOBILE	9	Telegram	9	Twitch: Live Game Streaming	9	Grubhub	9	Grindr
10	Call of Duty: Mobile	10	Helo	10	Tencent Video	10	Postmates	10	Skout
Travel		Shopping		Music & Audio		Health & Fitness		Finance	
1	Uber 142.5M	1	Wish 168.8M	1	Spotify 223.6M	1	Drink Water Reminder 40.5M	1	Google Pay 148.1M
2	Google Maps	2	AliExpress	2	YouTube Music	2	Mi Fit	2	PayPal
3	Booking.com	3	Amazon	3	Shazam	3	Flo Period & Ovulation Tracker	3	PhonePe
4	Grab	4	Pinduoduo	4	SoundCloud	4	Calm	4	Cash App
5	Google Earth	5	Club Factory	5	Drum Pad Machine	5	Samsung Health	5	Alipay
6	Airbnb	6	Lazada	6	Smule	6	Home Workout	6	Cloud Flash Payment
7	Where is my Train	7	SHEIN	7	JioMusic	7	MyFitnessPal	7	Venmo
8	Lyft	8	Mercado Libre	8	Gaana Music	8	Period Tracker Period Calendar	8	Personal Income Tax
9	Ola Cabs	9	Flipkart	9	Deezer	9	Six Pack in 30 Days	9	Sberbank
10	GOJEK	10	Shopee	10	Amazon Music	10	Fitbit	10	CASHIER

Future Services by whom ?

► Traditional Players (e.g. Telco's)

- Their business model: payed services from day 1, Interoperable and critical mass difficult.

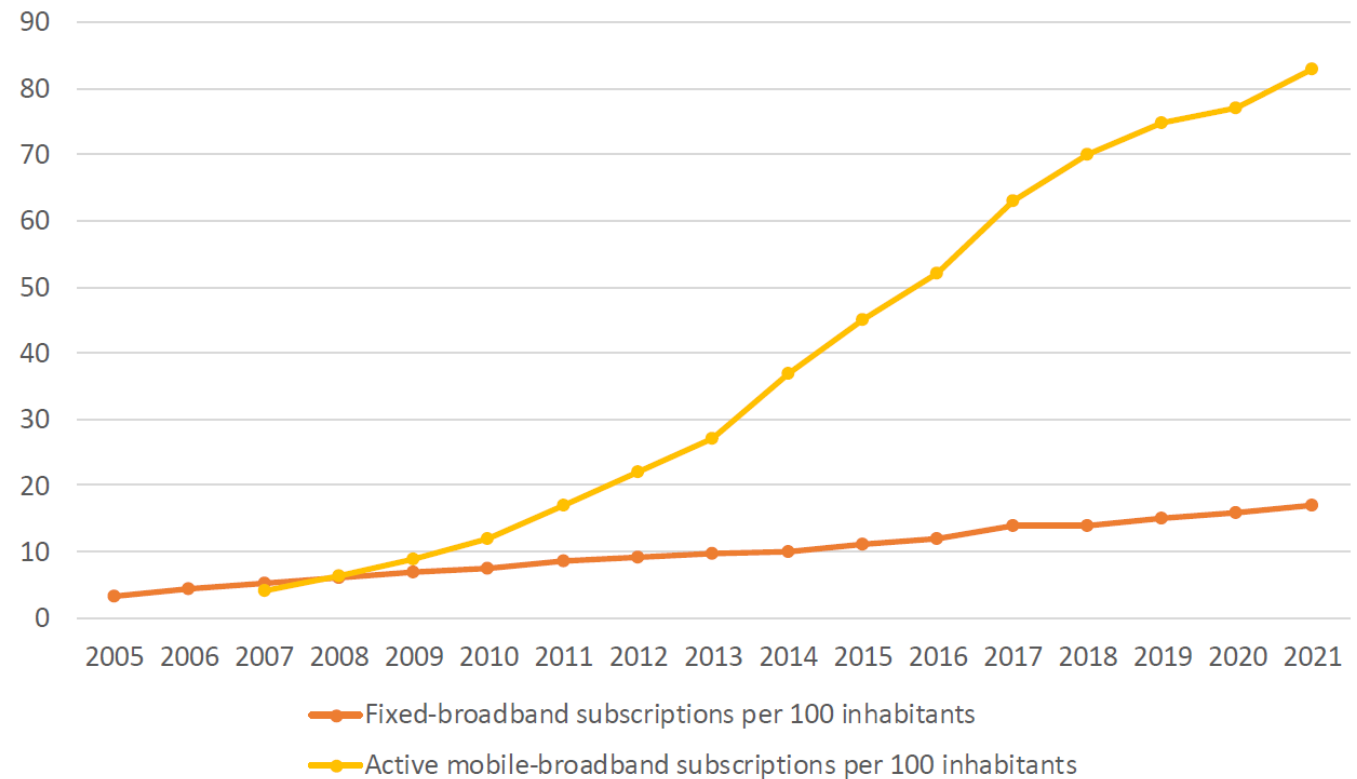
► Internet based companies (e.g. Google, FB)

- Their business model: Idea, trial and error, free first, ads further ahead, premium services.

► Startups

- Their business model: Idea, Trial and Error, mostly free, profit by selling company to the previous players.

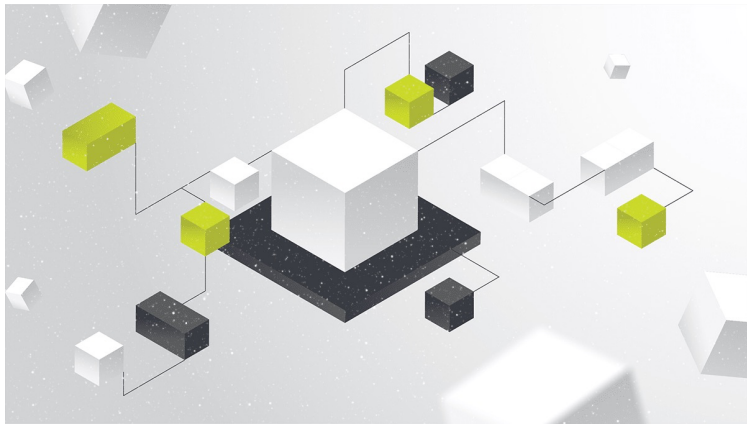
Broadband subscribers



How can that happen so quickly ?

- ▶ Large economies of scale (planet size)
- ▶ Access to ilimited resources (financial, computing, logistics)
- ▶ Rapid Application Development (RAD)
- ▶ **“Move Fast and Break Things”**
Mark Zuckerberg
- ▶ This means your application/service must be build to:
 - ▶ Withstand rapid growth (scale)
 - ▶ Be Agile (resilient to changes in use-cases)
 - ▶ Be developed by a large distributed team
- ▶ **Must be Engineered for growth**

The difference between Software Architecture and Software Design



- ▶ Software Architecture is the process of converting software characteristics such as flexibility, scalability, feasibility, reusability and security into a structured solution that meets the technical and the business expectations
- ▶ Software design is responsible for the code level design such as, what each module is doing, the classes scope, and the functions purpose.

Basic rules when developing software application

When you are developing a software application you should always obey the SOLID principles:

- ▶ **Single Responsibility Principle** means that each class has to have one single purpose, a responsibility and a reason to change.
- ▶ **Open Closed Principle:** a class should be open for extension but closed for modification. In simple words, you should be able to add more functionality to the class but do not edit current functions in a way that breaks existing code that uses it.
- ▶ **Liskov substitution principle:** this principle guides the developer to use inheritance in a way that will not break the application logic at any point. Thus, if a child class called “XyClass” inherits from a parent class “AbClass”, the child class shall not replicate a functionality of the parent class in a way that change the behavior parent class. So you can easily use the object of XyClass instead of the object of AbClass without breaking the application logic.
- ▶ **Interface Segregation Principle:** Simply, since a class can implement multiple interfaces, then structure your code in a way that a class will never be forced to implement a function that is not important to its purpose. So, categorize your interfaces.
- ▶ **Dependency Inversion Principle:** If you ever followed TDD for your application development, then you know how decoupling your code is important for testability and modularity. In other words, If a certain Class “ex: Purchase” depends on “Users” Class then the User object instantiation should come from outside the “Purchase” class.

What about if I'm developing a system?

- ▶ SOLID still applies, but your concerns are much broader, we are now discussing Architectural aspects:
 - ▶ Is my system extendable, modular and maintainable ?
 - ▶ Performance ?
 - ▶ Low fault tolerance, scalability and reliability
- ▶ You most often can't have everything: do tradeoffs...



In the following weeks

- ▶ We will learn about most important Software Architectures
 - ▶ SOA
 - ▶ MicroServices
 - ▶ Serverless
- ▶ Architectural Patterns
 - ▶ Message Oriented Middlewares
 - ▶ Enterprise Service Bus
- ▶ Role of these Technologies in the Internet and in 5G and beyond
- ▶ Discuss Cloud Computing and its role in this paradigm shift
- ▶ We will address how to manage software architectures
 - ▶ Containerization, Docker, k8s
 - ▶ Load Balancing and Redundancy
 - ▶ Monitoring
 - ▶ Storage



Planeamento

Semana	Mês	TP	4a
1	Fevereiro	Apresentação / Introdução / SOA	12
2	Fevereiro	SOA	19
3	Fevereiro	API's	26
4	Março	MicroServices	05
5	Março	Message Oriented Middlewares	12
6	Março	Enterprise Service Bus	19
7	Março	Serviços em Telecomunicações (3GPP, OMA) - SMS, MMS, IPTV, IMS, RCS, MANO.	26
8	Abril	Cloud Computing	02
9	Abril	Function as a Service	09
10	Abril	Infrastructure as Code: Docker	16
11	Maio	Infrastructure as Code: Kubernetes	07
12	Maio	Load Balancing and Redundancy	14
13	Maio	Infrastructure Monitoring	21
14	Maio	Storage	28
15	Junho	Apresentação Projecto Semestral	04