Introdução

Diogo Gomes dgomes@ua.pt

“Software is eating the world”

Marc Andreessen, 2011

- Every businness you can think off has either been digitalized or is in the process of become digitalized
- Larges 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: $126billion (statista.com)

amazon

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)

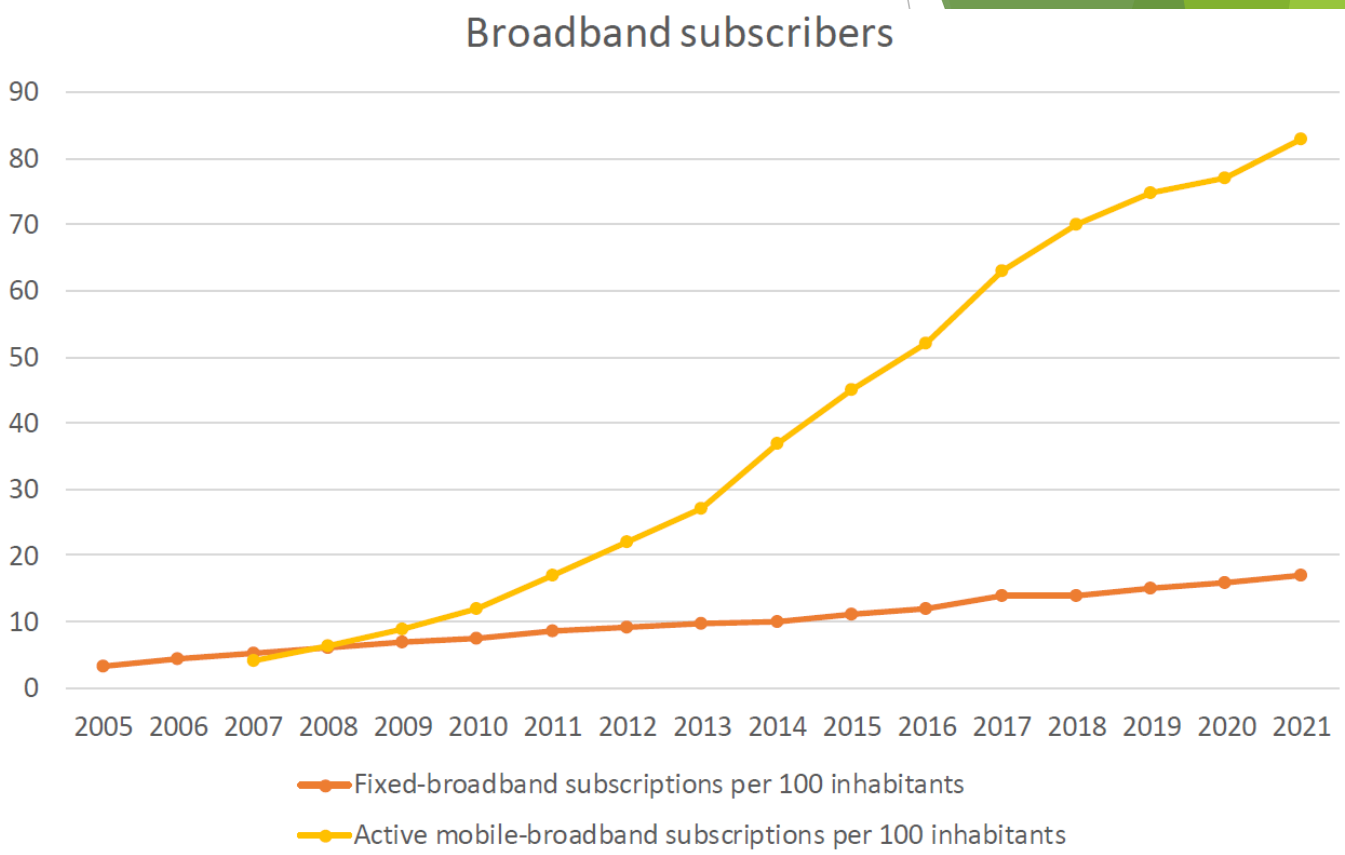
Most popular apps worldwide

2022, by category (million downloads)

Source: BusinessofApps

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.



How can that happen so quickly ?

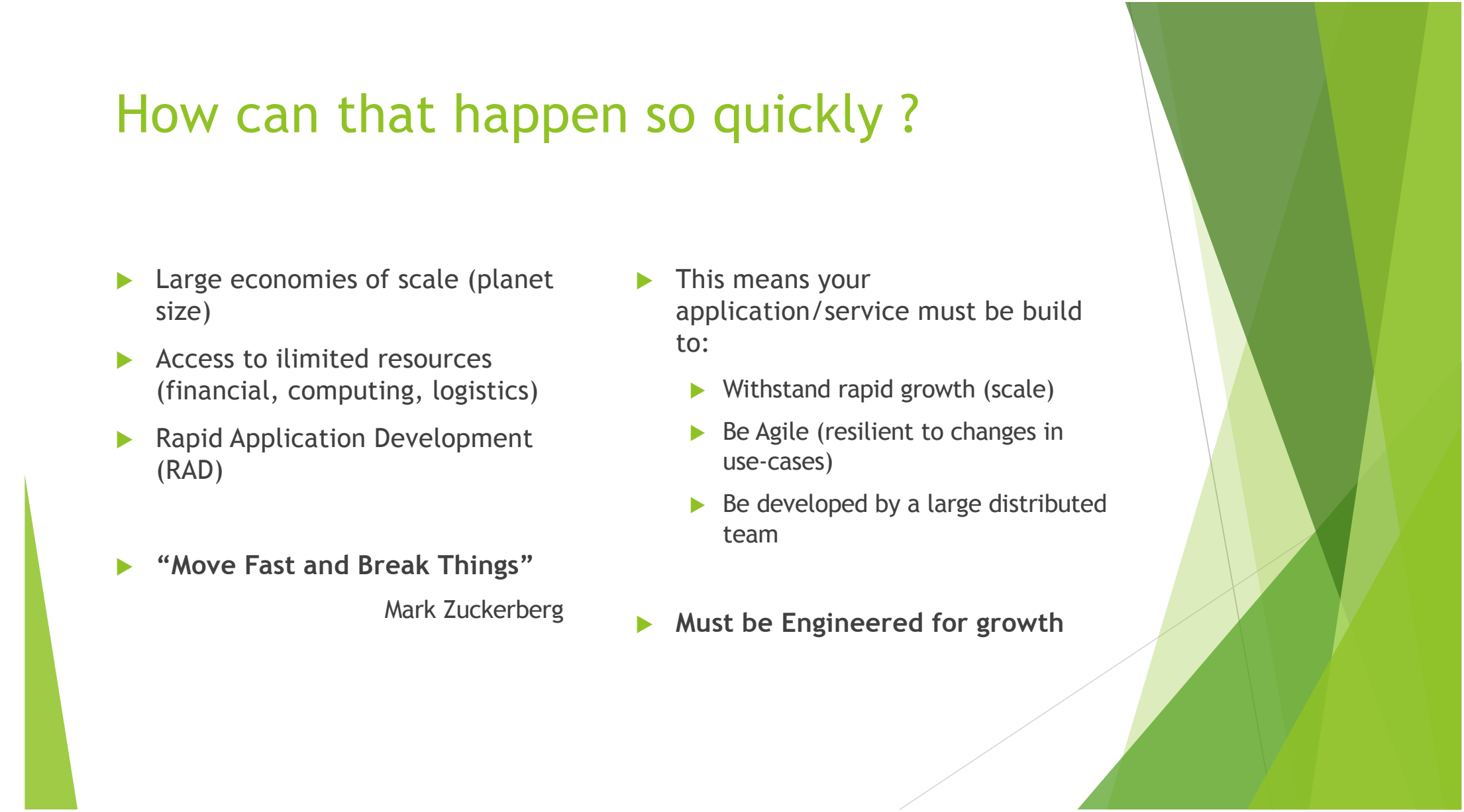
- Large economies of scale (planet size)
- Access to ilimited resources (financial, computing, logistics)
- Rapid Application Development (RAD)

- **“Move Fast and Break Things”**

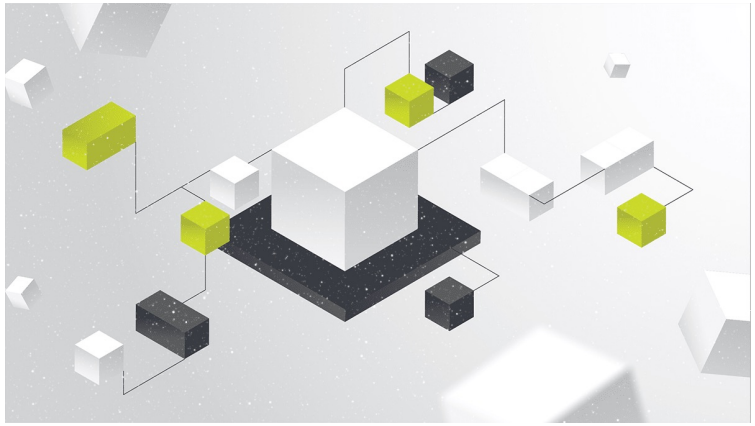
  Mark Zuckerberg

- 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 importante Software Architectures
  - SOA
  - MicroServices
  - Serverless
- Architectural Patterns
  - Message Oriented Middlewares
  - Enterprise Service Bus
- Role of these Technologies in the Internet and in 5G and beyhond
- Discuss Cloud Computing and it's role in this paradigma shift
- We will address how to manage software architectures
  - Containerization, Docker, k8s
  - Load Balancing and Redundancy
  - Monitoring
  - Storage