UNIVERSITÀ DEGLI STUDI
DI MILANO

Designing an Infrastructure (the Easy way)

Lesson I05

Dario Maggiorini (dario@di.unimi.it)
Online Game Design - A.A. 2021-2022

That Magic Moment When ...

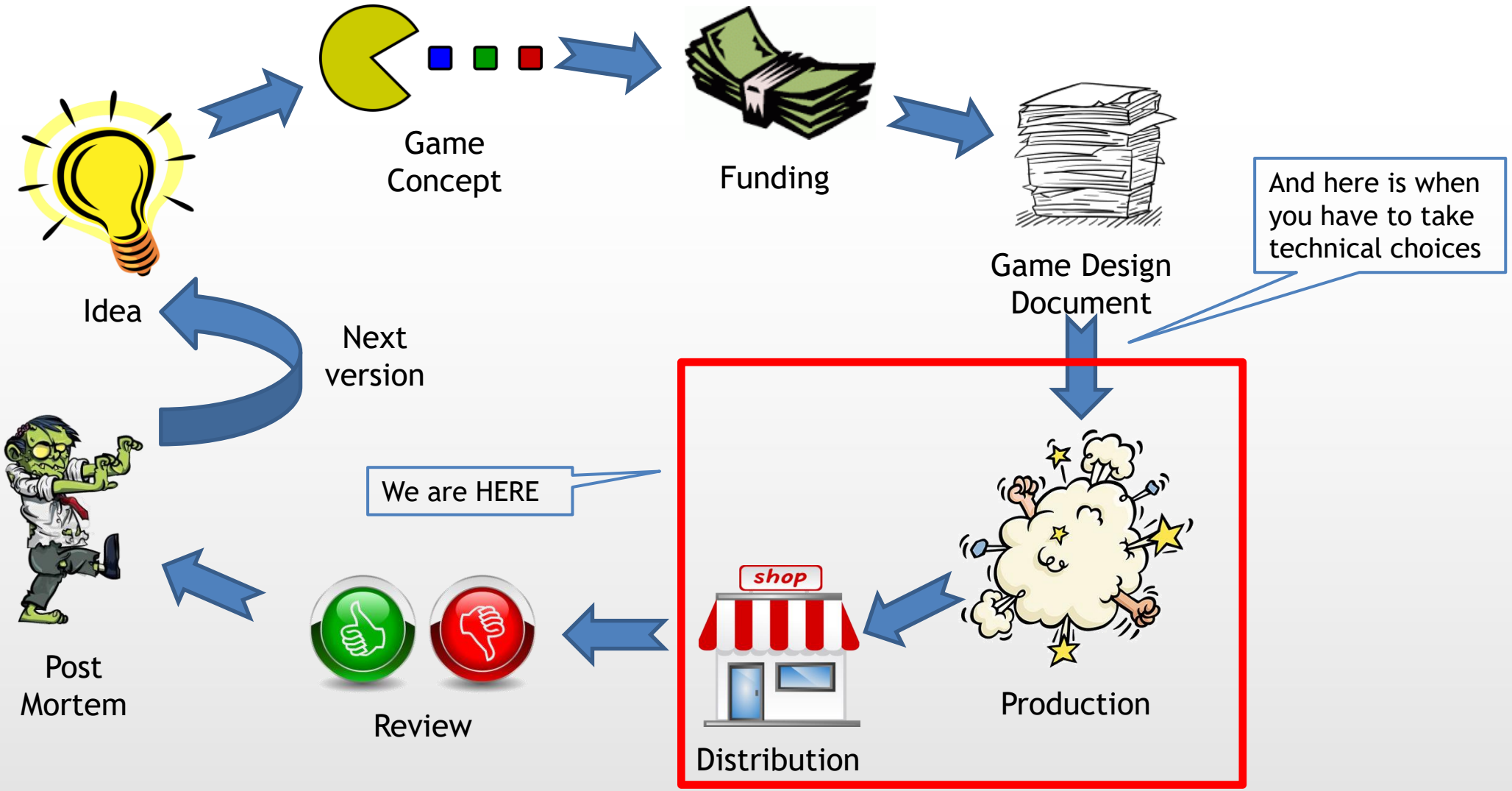
You must stop thinking about unicorns and rainbows (game design dream) ... and start managing:

- Hardware
- Software
  - Provisioning
  - Support
- Code
  - Design
  - Production
  - Backup
- Peoples
  - Salaries
  - Teamwork
    - Moral
- Business relations
  - Providers
  - Customer
- Space
  - Rent
  - Connection
  - Insurance
  - Supplies
    - Food
    - Power & Water

That Magic Moment When ...

... unless you are **ALREADY** a world-class superstar

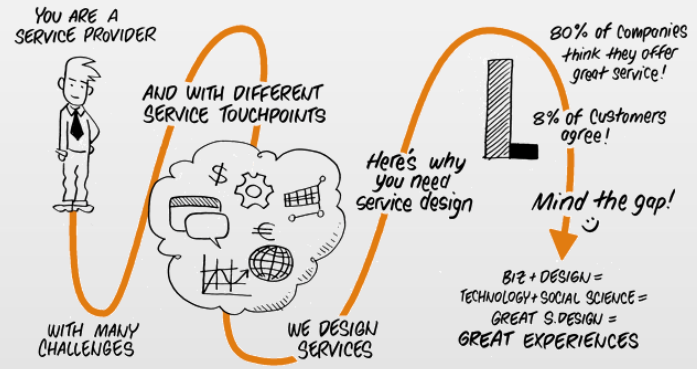
Lifecycle of a Game



Step 1: Understanding the Service

- You must understand exactly which kind of GAMING SERVICE you are going to provide
  - Describing it with a single sentence is a good way to go

- A service is usually described by:
  - What is going to be provided to the player
    - It should include the general idea of the genre of the game
  - What is your level of commitment to the game
    - The effort your company need to provide
  - The availability of the service
    - Usually 24/7 ... but it could change over time



Understanding Services

We need to provide service for a MOBA game where medium sized groups of independent people can fight in real time. The service will run 7/24 and offer seasonal events (with additional workload). We expect to host 250 millions subscribers worldwide and 80 millions active players monthly

We need to provide a service for an RTS card-based game to be run on mobile devices. The service will run 7/24 with season changing each month (with additional design effort). We expect to host 300 millions subscribers worldwide and 14 millions active players

Usage Numbers

- In order to keep the service operational, you must constantly monitor it

- Beta phase may give you some hints but there is no reliable way to foresee the future
  - Number of subscribed users
  - Number of active users
  - Daily maximum (peak) users

This is how you are supposed to be describing your potential workload

Forecasting Usage Numbers

- Once again, we are looking for a crystal ball

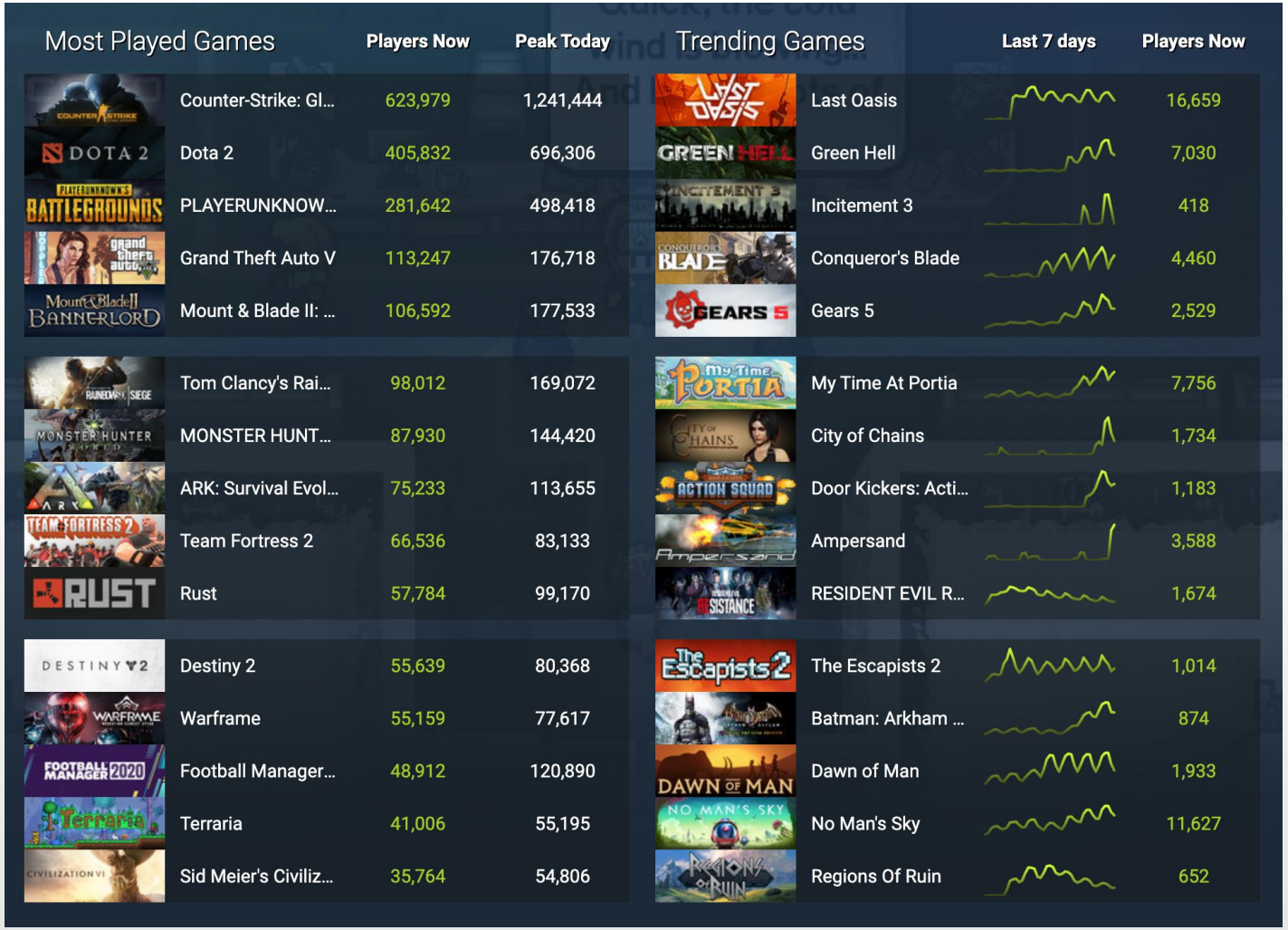
- But, this time, we DO HAVE a crystal ball

- You know the name of your competitors and you already investigated the market for similar games

  ... steamdb.info will give you hints

If you did not
... then you have a problem!

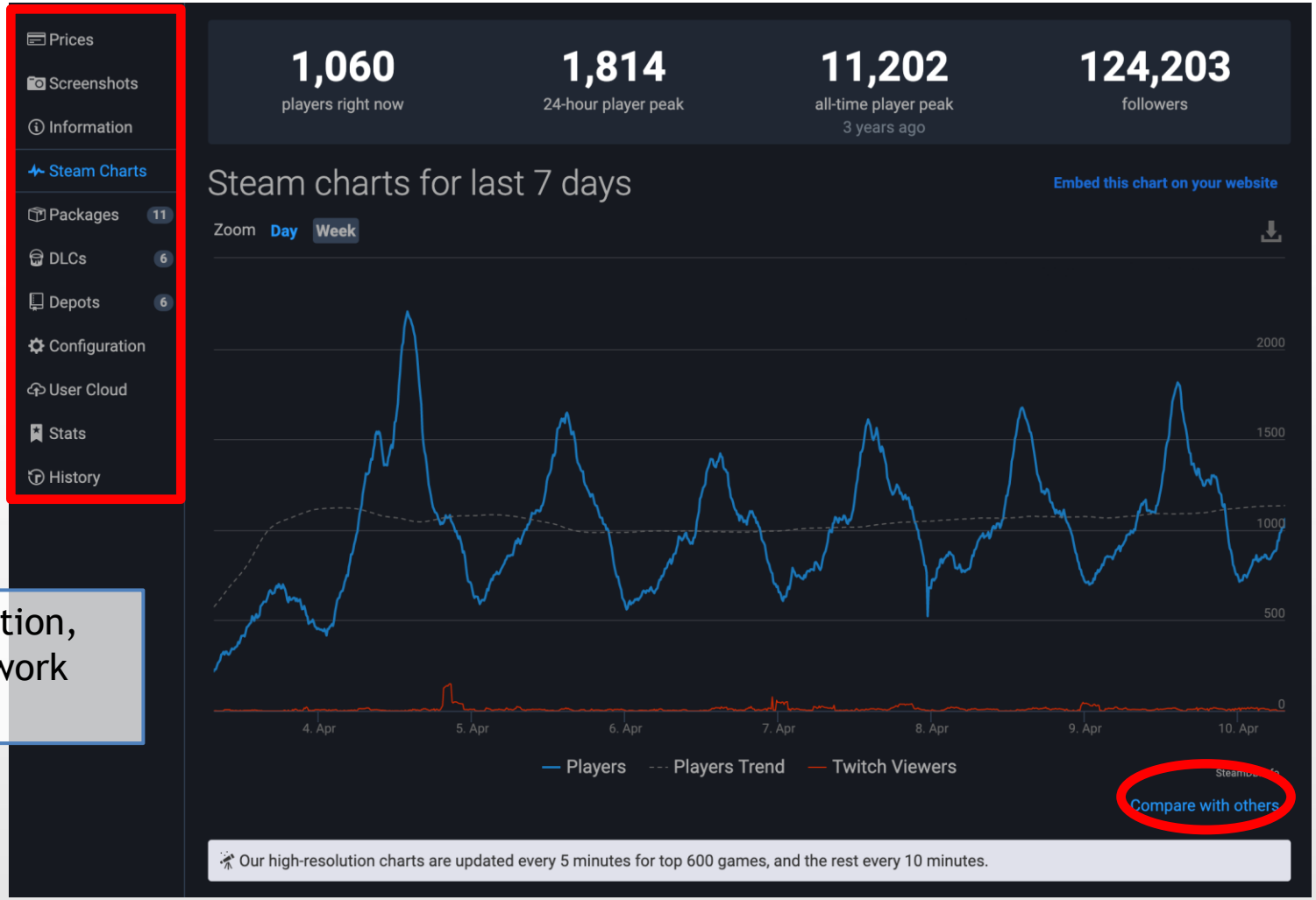
steamdb.info



Most Played Games	Players Now	Peak Today
Counter-Strike: Gl...	623,979	1,241,444
Dota 2	405,832	696,306
PLAYERUNKNOW...	281,642	498,418
Grand Theft Auto V	113,247	176,718
Mount & Blade II: ...	106,592	177,533
Tom Clancy's Rai...	98,012	169,072
MONSTER HUNT...	87,930	144,420
ARK: Survival Evol...	75,233	113,655
Team Fortress 2	66,536	83,133
Rust	57,784	99,170
Destiny 2	55,639	80,368
Warframe	55,159	77,617
Football Manager...	48,912	120,890
Terraria	41,006	55,195
Sid Meier's Civiliz...	35,764	54,806

Trending Games	Last 7 days	Players Now
Last Oasis		16,659
Green Hell		7,030
Incitement 3		418
Conqueror's Blade		4,460
Gears 5		2,529
My Time At Portia		7,756
City of Chains		1,734
Door Kickers: Acti...		1,183
Ampersand		3,588
RESIDENT EVIL R...		1,674
The Escapists 2		1,014
Batman: Arkham ...		874
Dawn of Man		1,933
No Man's Sky		11,627
Regions Of Ruin		652

steamdb.info

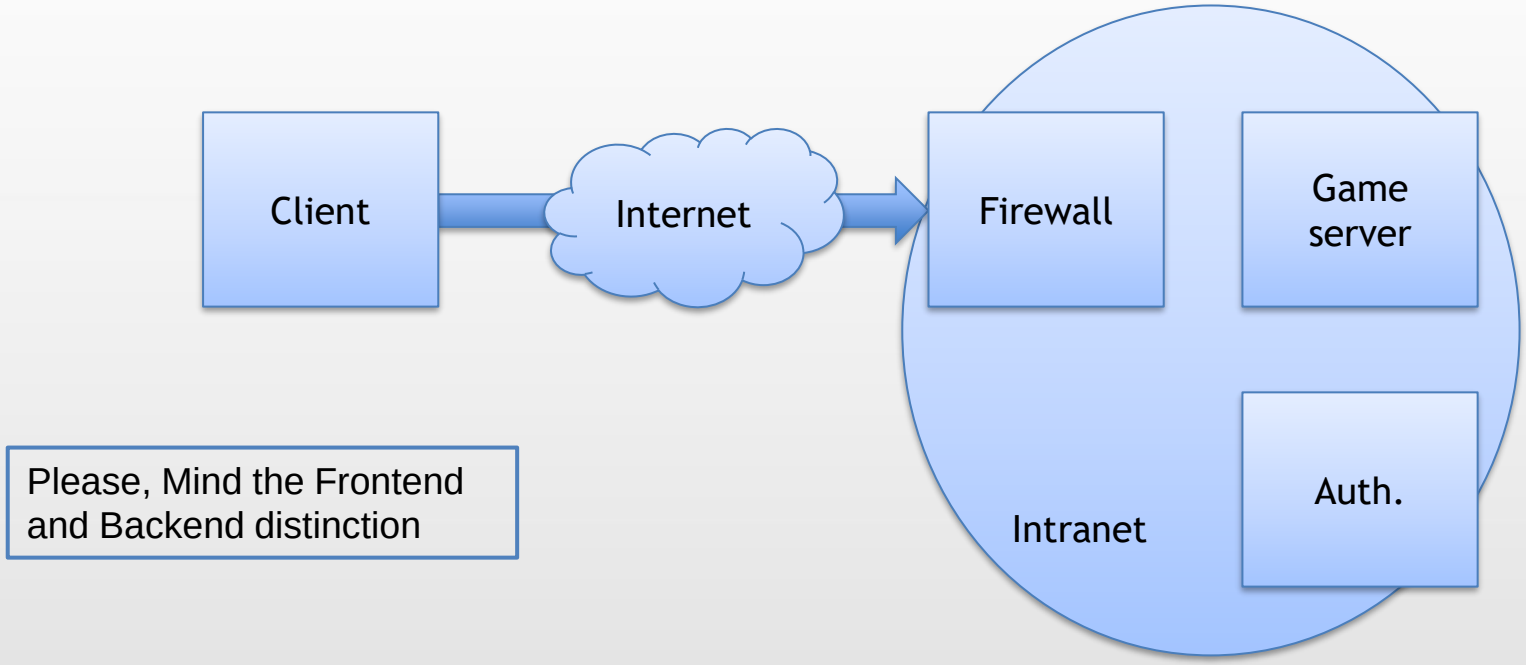


A LOT of useful information, not related only to network usage

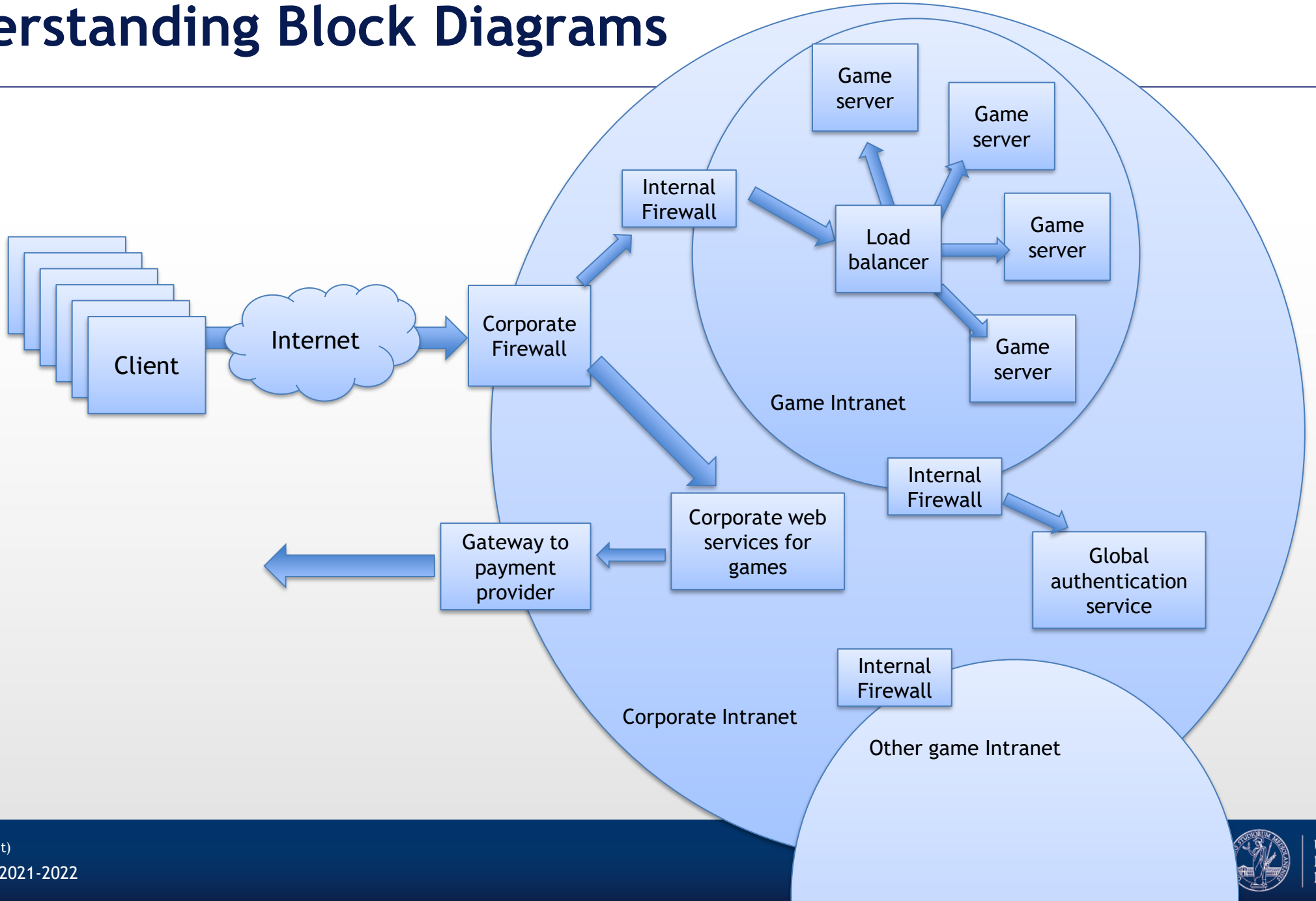
And you can also compare games

Step 2: Hi-Level Block Diagram

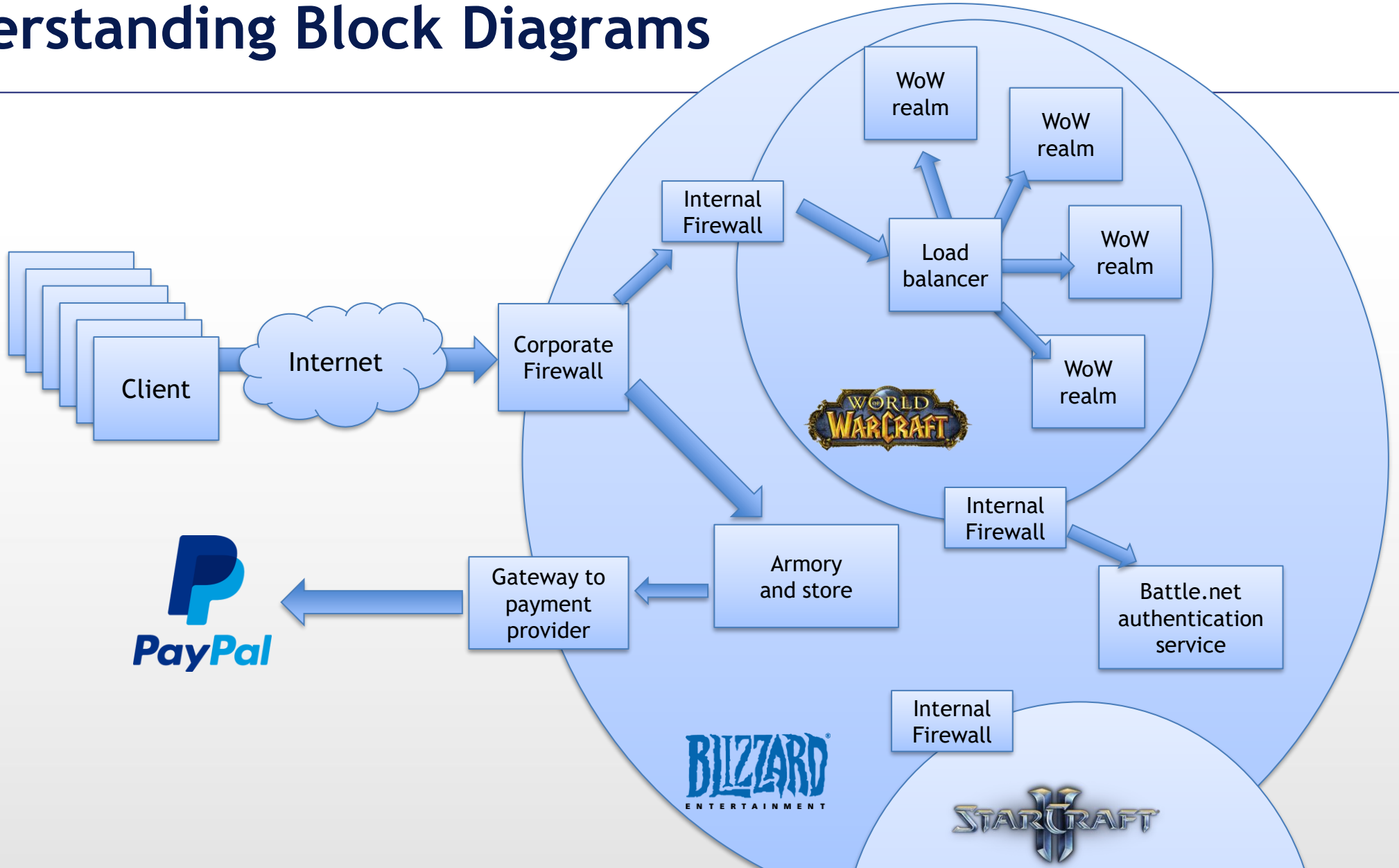
- You must lay down an hi-level block design to understand the data flow inside your infrastructure and what are the main functional component you need to set up
  - Focus on functionalities! Not hardware and software



Understanding Block Diagrams



Understanding Block Diagrams



Step 2: Common Mistakes

- Going too much into details
  - You must state that a functionality is available, not how that functionality is provided
    - No one care about how you authenticate your users.
      Just that there is some authentication system
    - No one care about how you make your users pay.
      Just that you have a way to get their money

- Forget about functionalities
  - Every box is a function or a sub-service in your architecture; never think to them in term of “I want that hardware” or “I want that software”
    - You need a FIREWALL! You do not need Cisco ASA
    - You need a WEB SERVER! You do not need Apache
    - You need a DATABASE! You do not need Oracle DB

My Boxes or Your Boxes?

- The mere fact that you need a functionality does not imply that you must hold that functionality in your network
- Each one of the boxes you put in your schema may be:
  - A service you provide
    - This is the “classical” approach: you buy a server and run your service using your hardware
    - We can call this “system on premise”
  - A service you get from a provider over the Internet
    - This is the “new” approach: you pay someone to rent a service, usually in a cloud
    - We can call this “cloud gaming”
- NOTE: this decision can be taken for each single box
  - So, you can have a mixed environment where you manage internal core services while paying for everything is left
- Golden rule: you always try to pick the cheapest solution!

Wow! .... This was really unexpected!

My Boxes

- Albeit “old style”, there are still many reasons to keep your system on premise
  1. You already have the hardware
     - Or someone already committed to buy it (sh*t happens)
  2. You work for a huge corporation that already has all the resources and organization
     - That could be a huge “standard” datacenter or an internal cloud
  3. You manage data or IPs that you do not want to be in someone other’s infrastructure

- Then, you must account also for specific costs and activities
  1. Hardware maintenance and periodic refresh
  2. O.S. Maintenance and updates
  3. Backup
  4. Power and ventilation
  5. A (guarded) place where to put everything

Your Boxes

- This is the favorite approach today
  - Because it **looks** cheaper than having the systems on premise
    - Maybe it is, but better remember the content in the lesson about cloud and gaming

- Of course there are many advantages
  - No need to buy hardware
  - No need to ensure/maintain/backup anything
  - No need to power/store/cool anything

- But ... there is a catch!

Your Boxes, Your Catch

- There is no easy way to understand a priory how much you are going to pay for the service
  - Tables are sometimes confused
  - There is no way to compare providers
  - Your bill will depend on the number of concurrent players
    - Unfortunately, that is very difficult to estimate
- All you can do is pick one and see
  - But remember, it is a blood shed to change provider during production

- On the bright side, the average lifetime of an online game is shorter than three years. So, you can at least use their software to have a better understanding of your saving perspective

Boxes in the Cloud

- Under the proposed perspective, cloud services for games are just an externalization of game-related services (and hardware)
  - Yes! We do not need to run the service ... but we also do not need to own the infrastructure

- At this point, what we need to know for our planning is:
  1. Which gaming cloud providers are out there
  2. What are they offering
  3. How to use (and interface) their services
  4. And, of course, how much do they charge :)

Always-on-Cloud Stuff

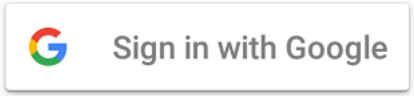
- Whenever you must manage data that either:
  - Is private to the user
  - Requires management on your side
- trust me, **you do not want it in your database**!

- Private data
  - You do not want his/her credit card number
    - If the number get stolen (no matter where) you will be liable
    - Much better outsource to paypal, even if they ask for a fee!
  - You do not want his/her data covered by privacy
    - Otherwise, you must comply to all DGPR rules, and that is expensive

- Managed data
  - You do not want his/her password
    - Because you must give support when it is stolen or forgotten
    - Much better use an authentication provider or authenticate via something which is not a password

About Cloud Authentication

- Authentication provider
  - This is a trusted authority that will give you half of a crypto cookie explaining that "you can trust the owner of the other half of this cookie is Mr. X"
    - Then ... we can discuss about the definition of "trust"



  - "Light" authentication providers:
    - They just make sure you are "the same guy as the last time"
  - "Formal" authentication providers:
    - They make sure you is really YOU

- Using something which is not a password, usually something owned or connected to the player
  - It may be a phone
  - It may be a phone number
  - It may be a fingerprint

Step 3: Hardware

- Now ... After we have a clear set of functionalities, we must understand which is the best hardware to buy in order to fill the boxes that we
decided to keep on premise

- There is no actually “right”
or “wrong” way to do this
  - Unless, you account as “wrong”
the fact you are using too much money
    - Your Boss will do, be prepared :)

Step 3: Common Problems

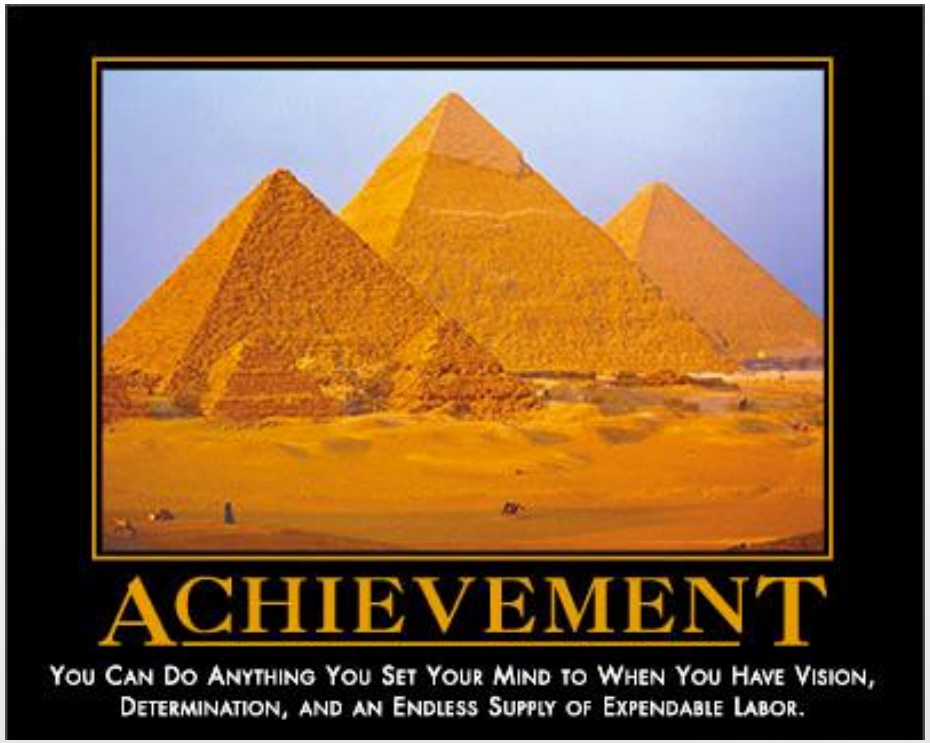
- Trying to have a specific hardware vendor is NOT (always) a mistake, for many reasons
  - You already have expertise about the products
  - You are already a customer, so you can have a good deal
  - There is a collaboration ongoing
  - You know the hardware is expensive, but support is much better than the competitors
    - Price is a relative concept, after all
    - You must think in term of TCO (Total Cost of Ownership) rather than looking to each single invoice
  - The rest of your infrastructure uses the same hardware
    - Compatibility matters ... A LOT!

- Of course, “trying” means also some elasticity
  - If you find an extraordinarily good deal
  - If you work for a corporate, you usually want to put your hardware providers into competition to get better prices

- As for the cloud, it is very difficult to find actual prices for business-level hardware
  - A sales representative must be contacted, and then he/she will start pestering you night and day for money

Step 4: Services

- In this step, you must understand and list every software service you are going to get from third parties
  - This is including cloud services
- This is very like software, just you have nothing to install
  - All the previous recommendation (and mistakes) still holds



Step π: Budget

- This is not fitting in the list as a number because there is no ordinal point where to put it
  - Should be at the beginning, because it is a constraint you use to take all decisions
  - Should be at the end, because if you have money left you can backtrack and increase some resources
- Usually, in real life, it can be elastic
  - Subject to negotiation
  - May change over time (for bad, usually)

- No matter what you do,
  in real life you will never get as many money as you wish

About Your Project

- You are going to produce a Technical Design Document

- In the TDD, you must apply the concepts you learned today
  - Including the outline of the architecture
  - Including all the lists for the required hardware and services

Ideally, it should be a document you can give to another department of your company and ask “please, do this for me"

About the Technical Design Document

- The TDD must describe in detail your plan of development to bring your project to a PUBLIC BETA
- The TDD is NOT describing what you are going to present for the exam or, eventually, in the NGD event.
  What you are going to make for the exam is a demo, which can be SMALLER and LESS FEATURE-COMPLETE than the envisioned public beta
- There will be a template available for the TDD. Look in the other attachments to this lecture
  - Anyway, it is a template. It can (and should) be modified based on your project's requirements

About the Money

- Once you have completed your TDD, you should at least make an effort to estimate the project cost

- There will be no right or wrong number. Use your judgement to understand if it is reasonable
  - e.g., asking for one million euros for an indie mobile game is definitely **not** reasonable

Study Material