UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Programrendszerek fejlesztése
Bevezető

Dr. Bilicki Vilmos

Szegedi Tudományegyetem

Informatikai Tanszékcsoport

Szoftverfejlesztés Tanszék

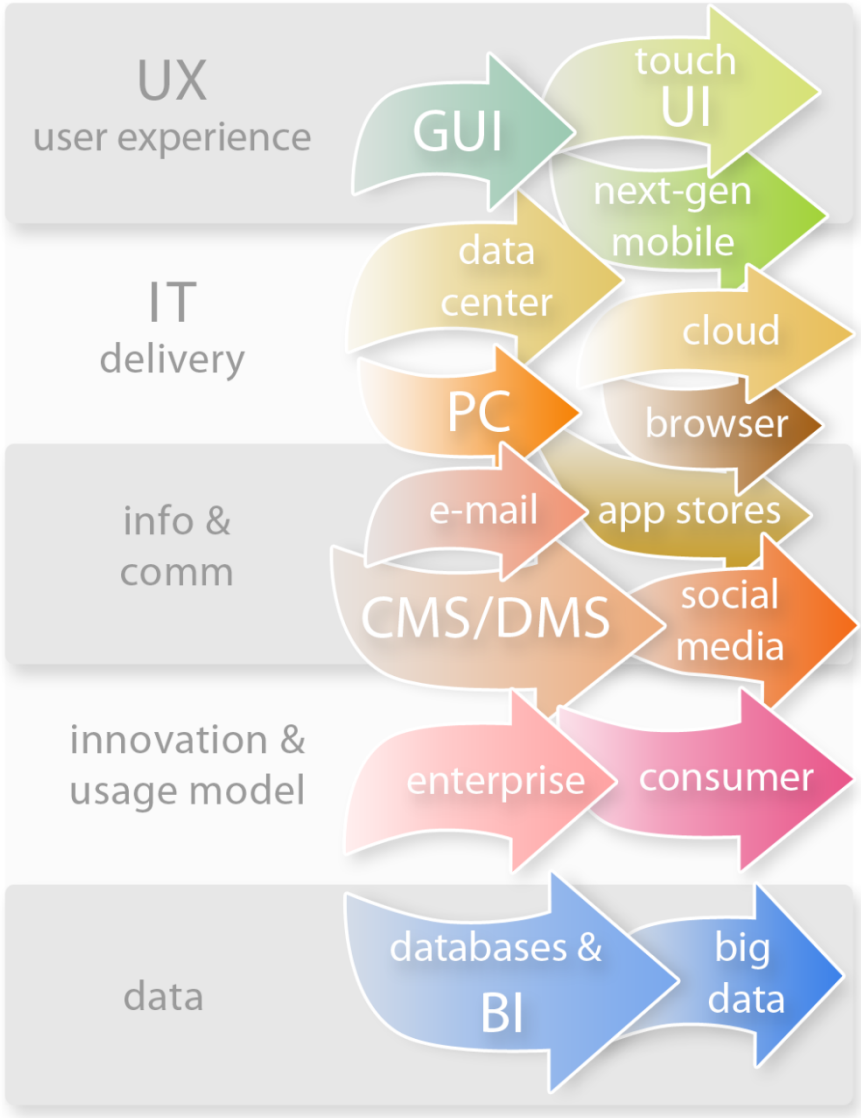
Tartalom

- Követelmények
- Bemutatkozás
- Célok
  - Fókusz a
    - Nem funkcionális/rendszer követelmények
    - Skálázhatóság
- A félév áttekintése

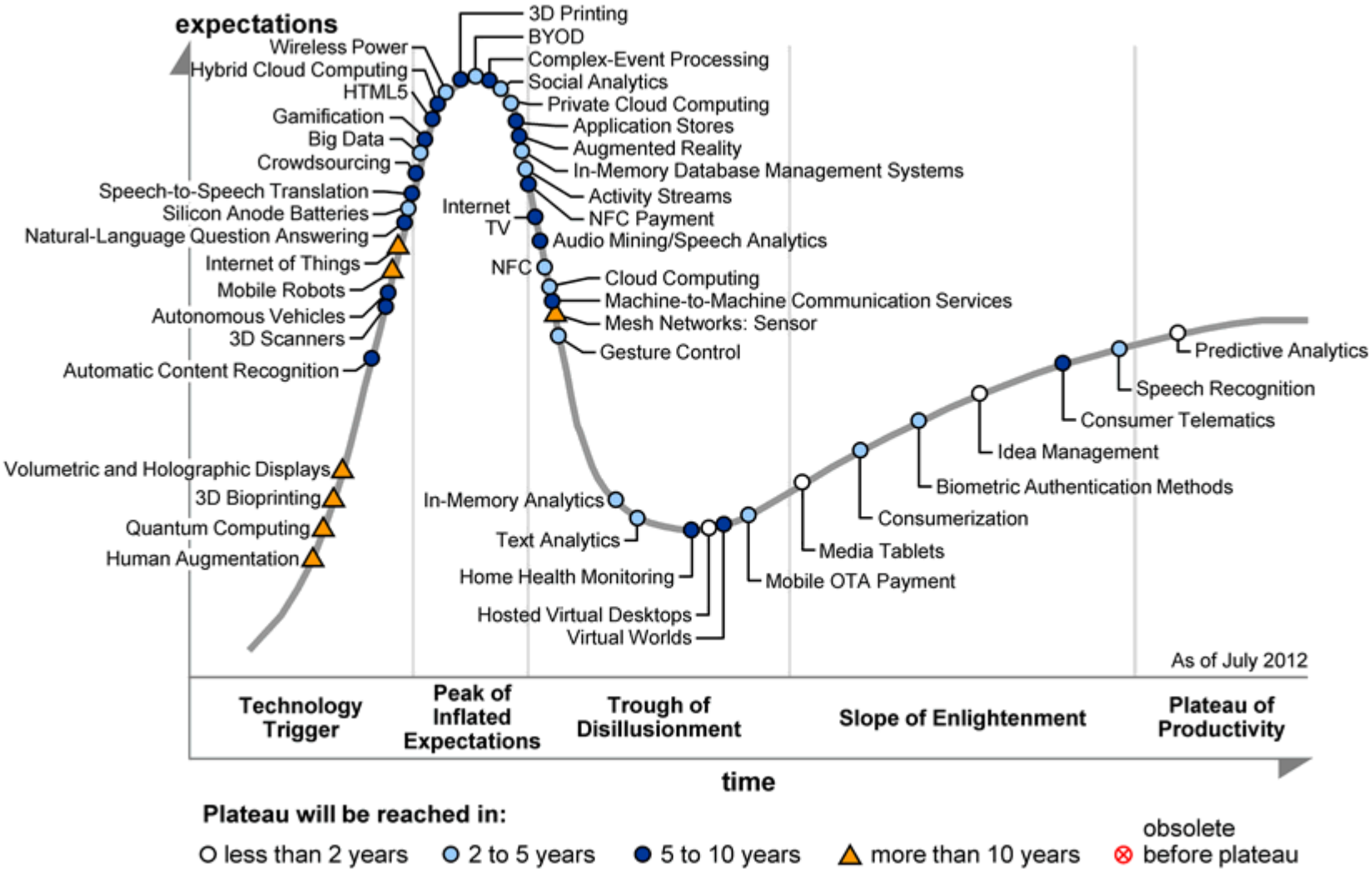
Követelmények

- ZH előadáson 50% (20%) (2019.04.16-i előadáson)
- Vizsga az osztályzat 80%-át adja

Trendek

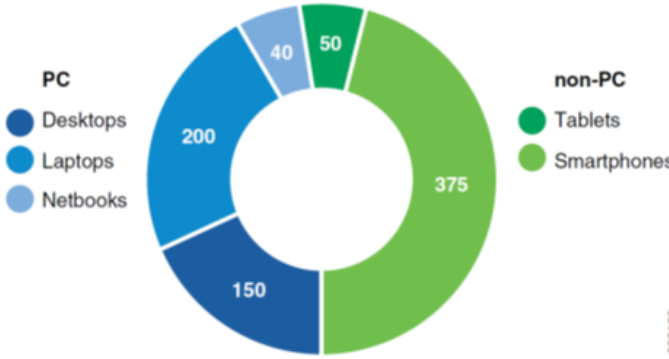

UX
user experience
GUI
touch UI
next-gen mobile
IT
delivery
data center
cloud
PC
browser
info & comm
e-mail
app stores
CMS/DMS
social media
innovation & usage model
enterprise
consumer
data
databases & BI
big data

Gartner: technológia fejlődés



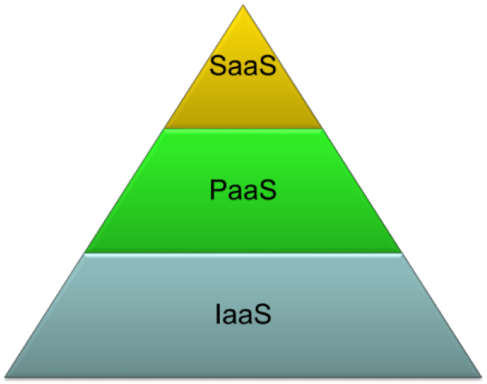
Paradigmaváltás: Mobil eszközök

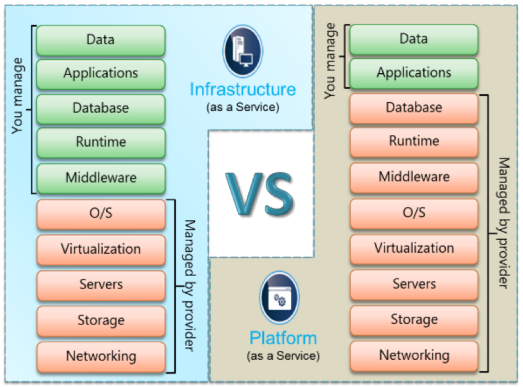
- Tények:
  - 6,7 Milliárd mobil előfizető, 1.3 Milliárd okos telefon
  - 1,3 Milliárd 3G kapcsolaton, 1.1 Milliárd WiFi keresztül kapcsolódik az Internethez
  - A leggyorsabban növő iparág az emberiség történetében
  - Az eszközök 2/3-án van kamera, a felhasználók 72%-a használja
  - Az emberek naponta 40x nézik meg a telefonjukat
  - A mobil tulajdonosok 91%-a 7x24 órában a telefonja közelében van
  - Az USA tévénézők 88%-a nézi a telefont is tévézés közben
  - Az USA vásárlók 49%-a változtatta meg vásárlási szándékát a boltban a telefonon szerzett információk miatt
  - Minden harmadik ember olvas híreket a telefonján
  - Több mint 1 Milliárd ember:
    - Tölt le alkalmazásokat
    - Játszik a telefonon
    - Használja a szociális háló alapú alkalmazásokra
- Előrejelzések:
  - 2014 az Internet használók fele sohasem használt PC-t
  - 2014 az eladott telefonok fele érintőképernyős
  - 2014 Több az előfizetés mint amennyi ember él a földön

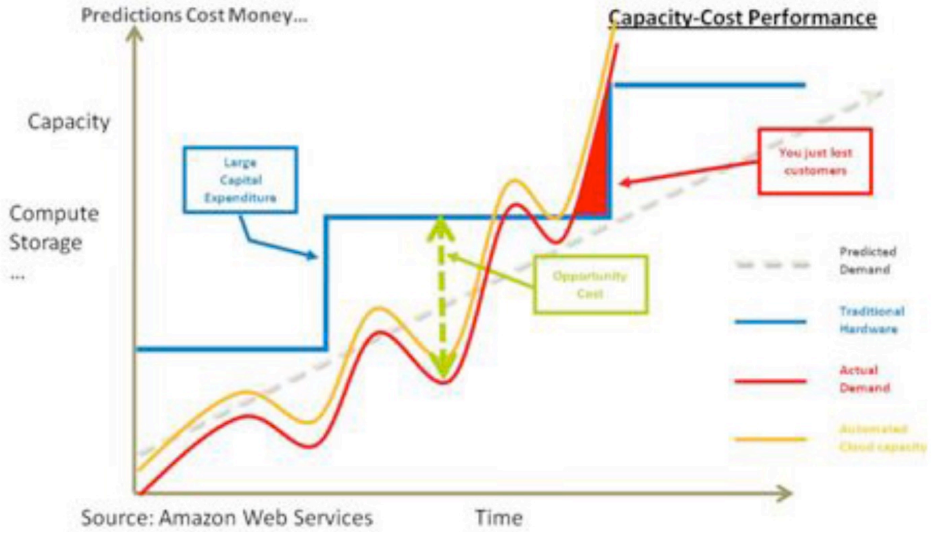


Paradigmaváltás: Virtualizáció

- Irány:
  - IT kiszervezés
  - IT mint szolgáltatás
- Felhő:
  - Robosztus
  - Skálázható
  - Hely független
  - Biztonságos
  - Olcsó

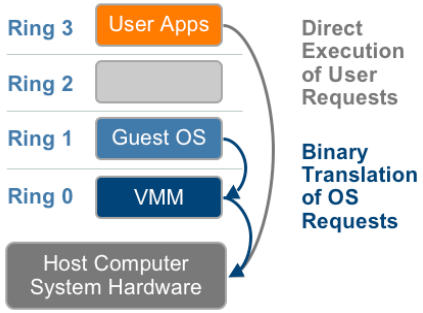


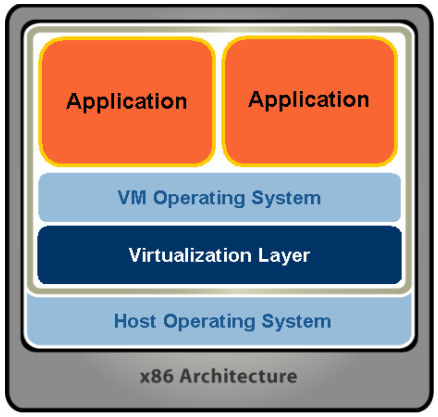


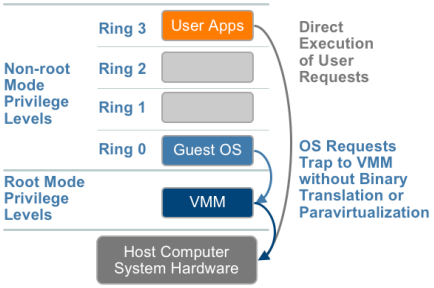


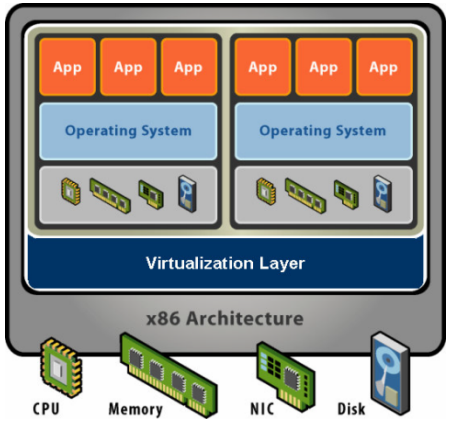
Típusai

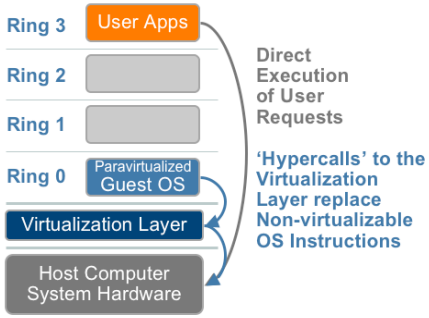
- Type 2: Hosztolt architektúra:
  - Befogadó Oprendszer felett van futtatva
  - Kicsi kontextus váltó meghajtó
- Type 1: Csupasz fém architektúra
  - A Hypervisort közvetlenül a HW-re telepítik
- Paravirtualizáció
  - Az operációs rendszer tud a Hypervisorról





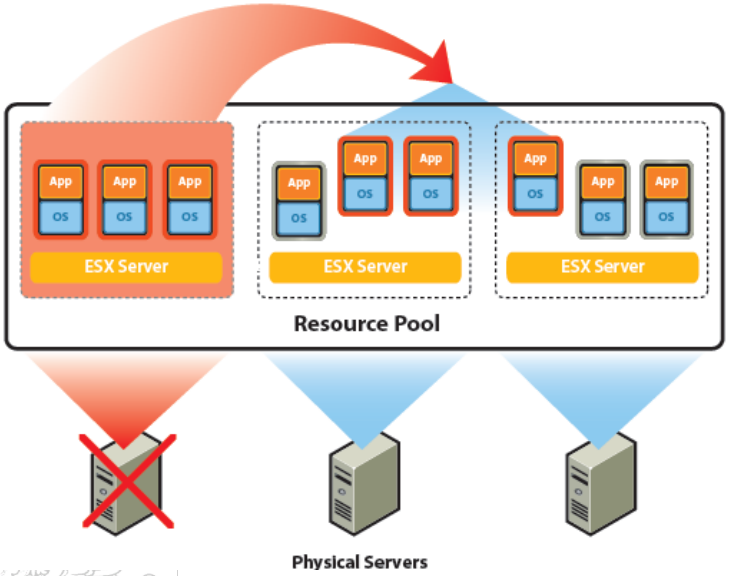




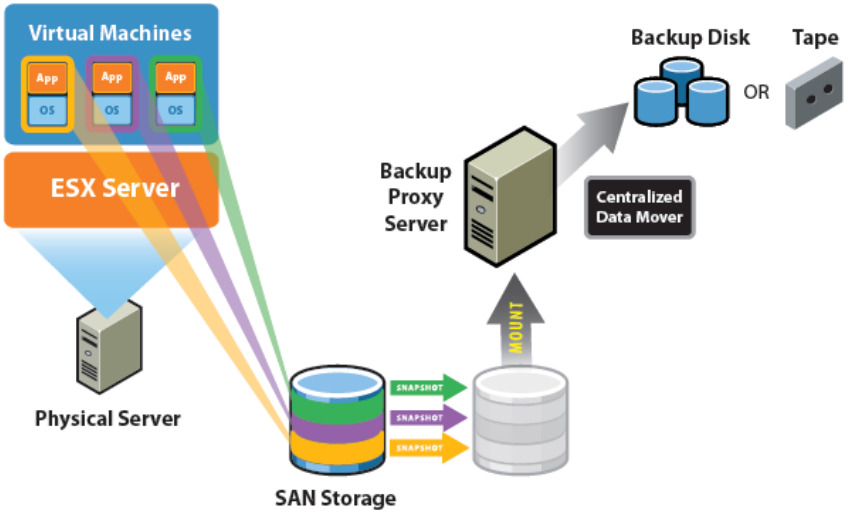


Virtualizáció: Képeségek

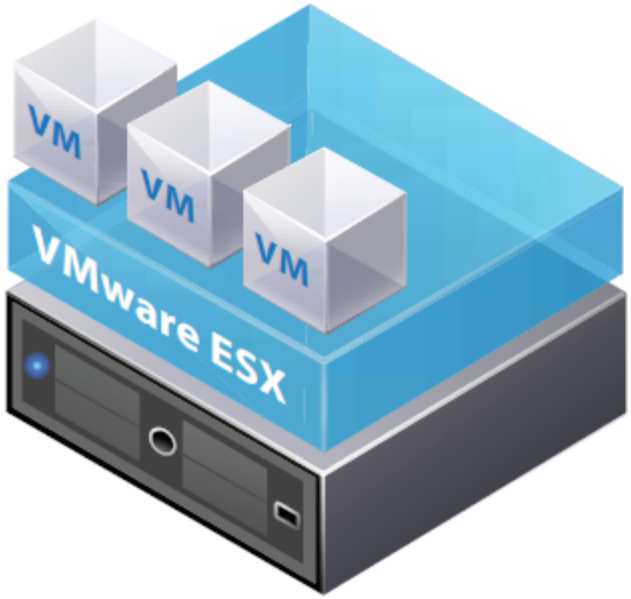
Magas rendelkezésre állás

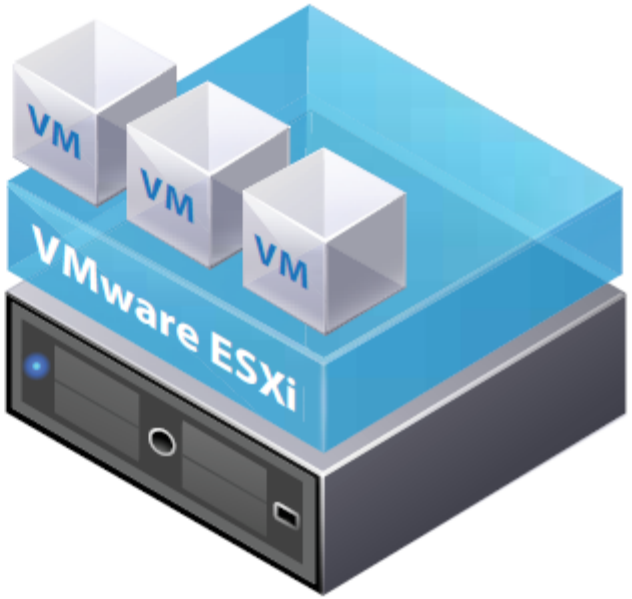


Konszolidált mentés

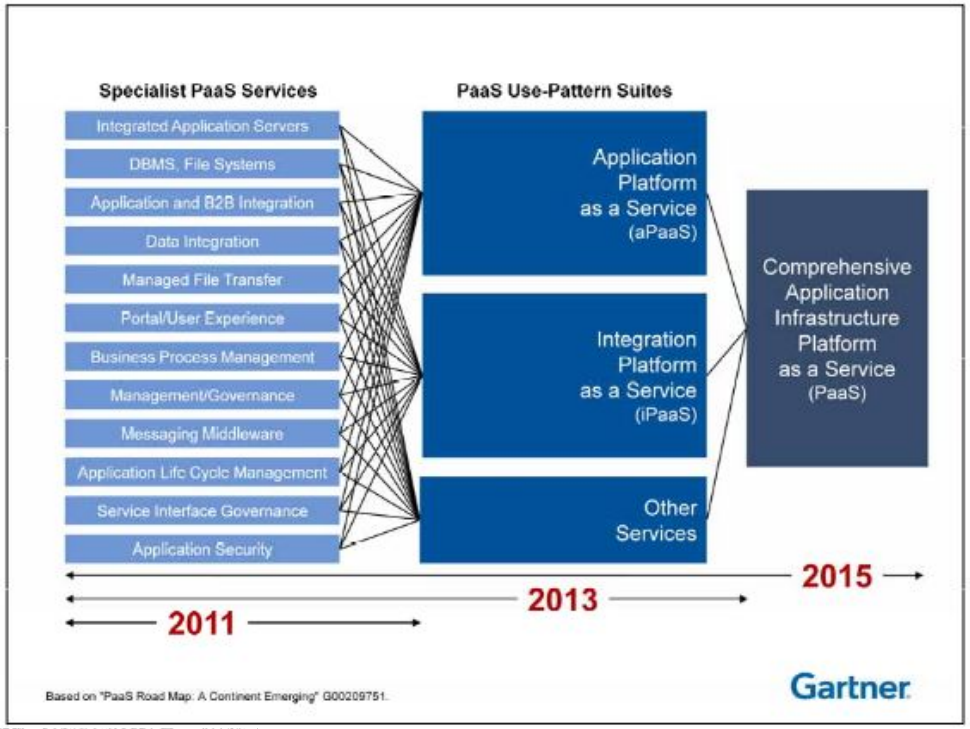


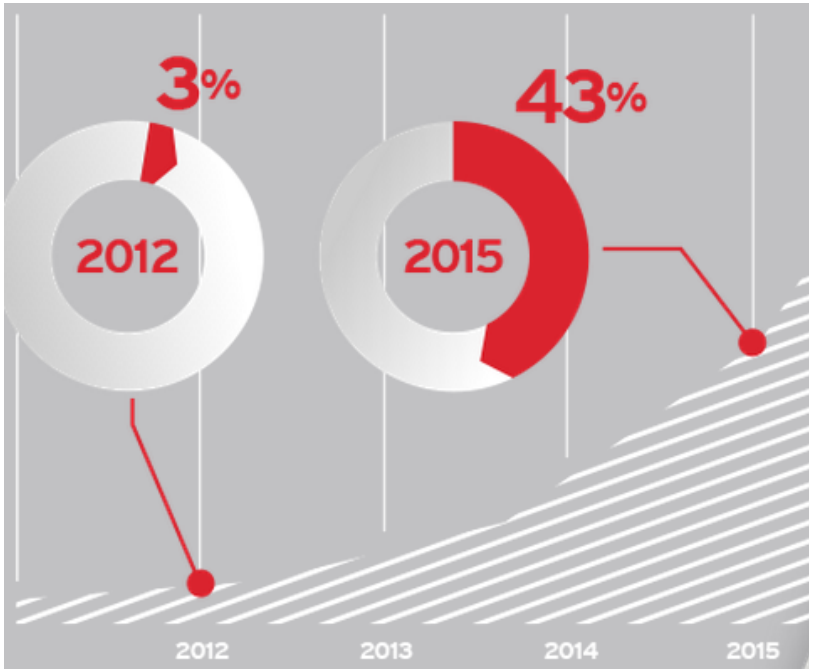
Probléma:





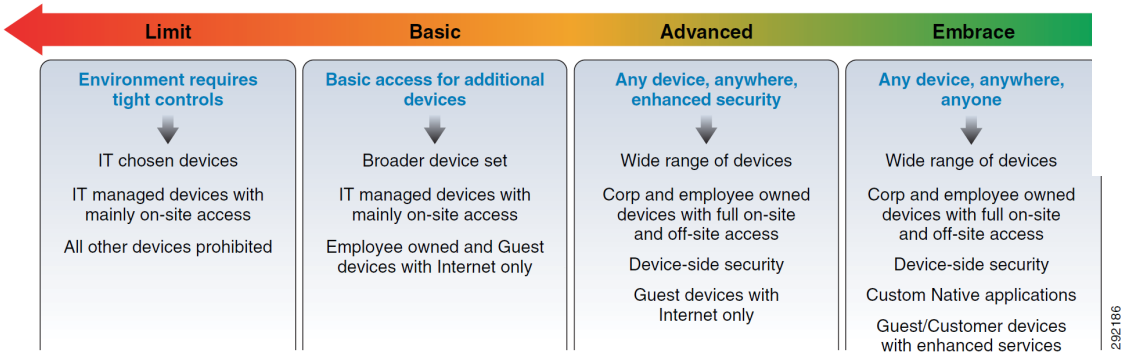
PaaS - Gartner





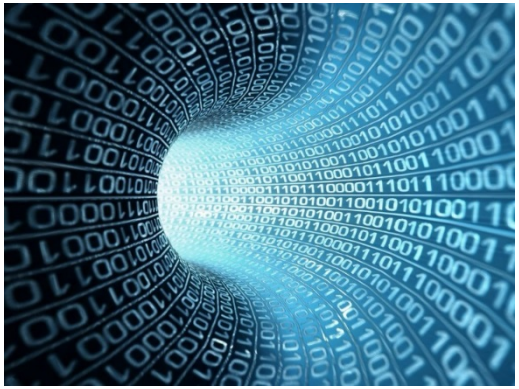
Paradigmaváltás: BYOD/BYOA

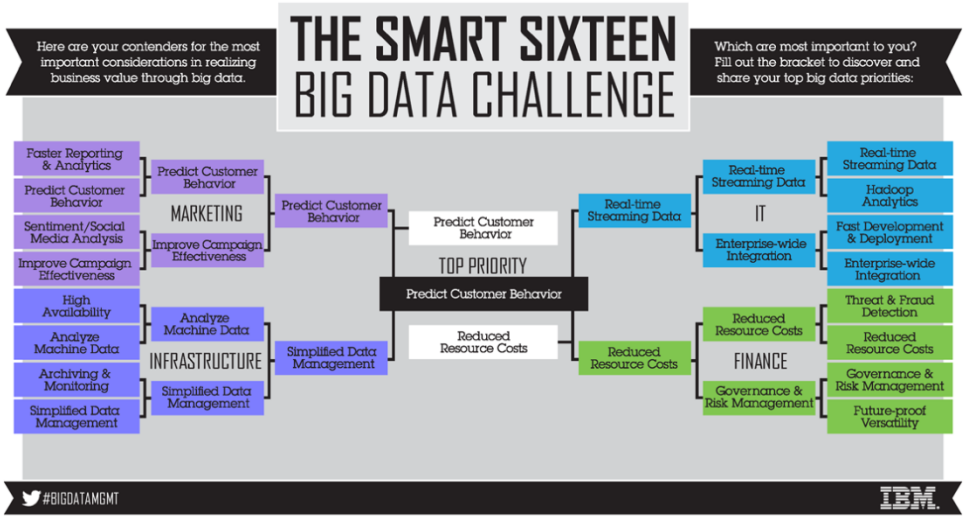
- A munkaidő és a szabadidő nem különül el
- A dolgozók produktívabbak a saját környezetükben
- A dolgozók igénylik a saját környezetet

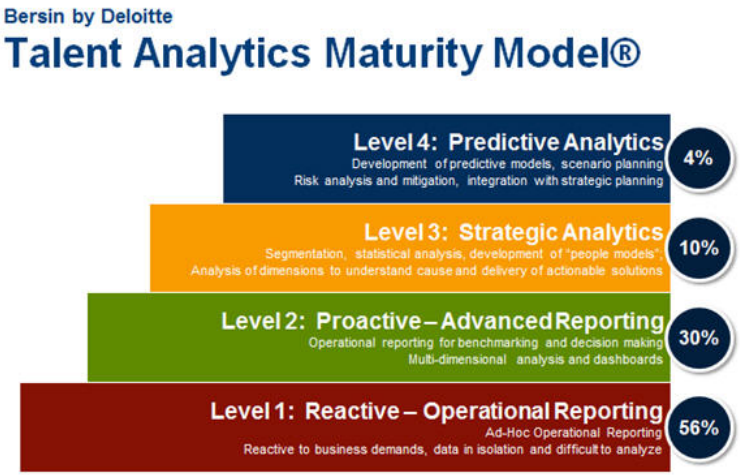


Paradigmaváltás: Nagy Adat

- A technológia, infrastruktúra adott a nagy adatok kezeléséhez
- Adat -> Érték

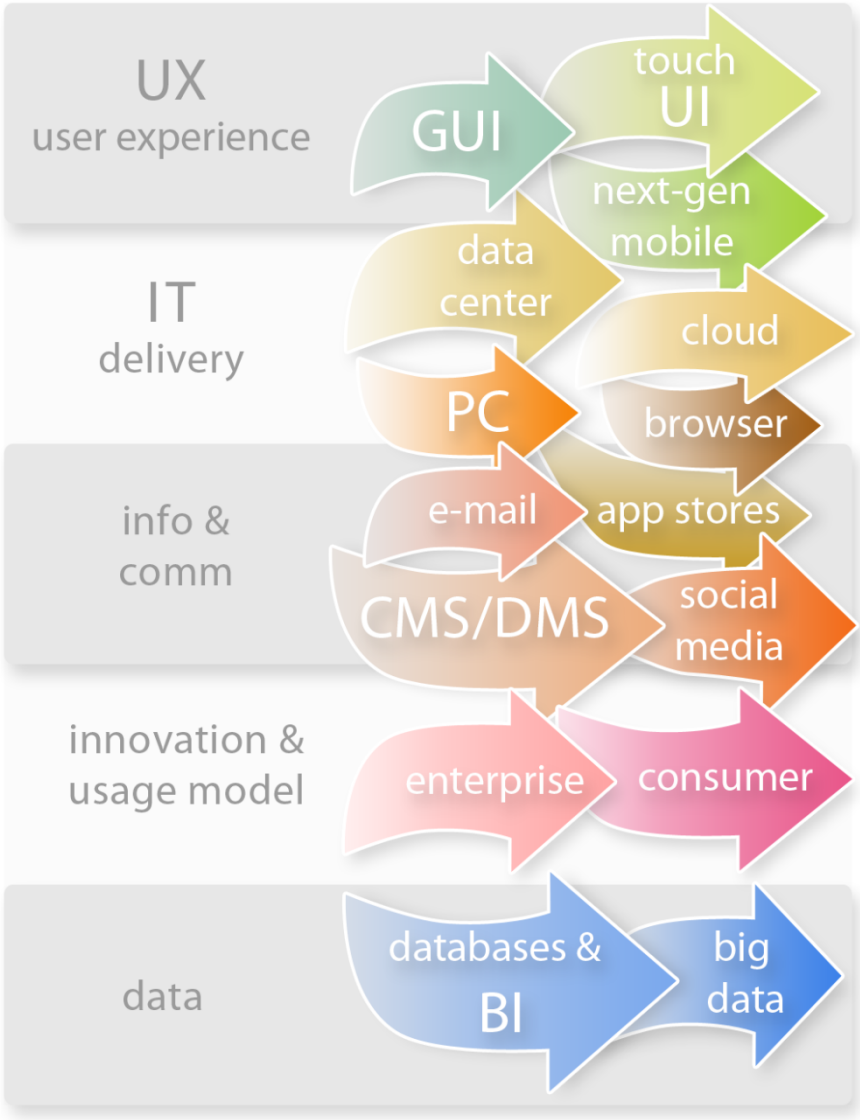






Összefoglaló

- A virtualizáció alkalmazása ma már a legtöbb helyen elfogadott, bevett gyakorlat
- A felhő ugyan komoly lehetőségeket biztosít, de a biztonság nyitott (egyre nyitottabb) kérdés
- A mobil az alapértelmezett platform



Célok

- Startégiai szintű áttekintés:
- Nagy skálázható rendszerek fejlesztése
  - Problémák
  - JEE alapú megoldások
    - Milyen rétegeket használjunk?
    - Hogyan fejlesszünk produktívan skálázható alkalmazást?
  - JavaScript/Typescript alapú megoldások
    - GUI vs Backend
    - Multiplatform

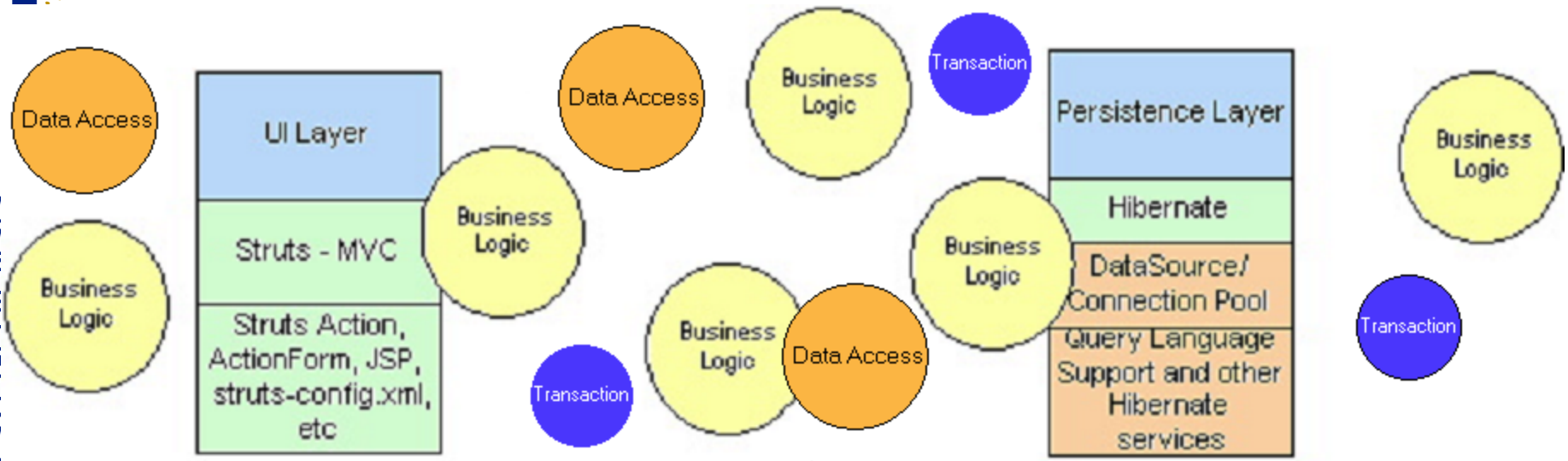
A tantárgy tematikája

- Trendek
- Architektúra
  - Felhő, virtualizáció
  - Mikroszolgáltatások
  - Web vs- Mobil
- TypeScript
- Elosztott rendszer
  - Skálázhatóság, architektúrák, CAP
  - REST/WS
- Adatkezelés
  - NoSQL
  - MongoDB
- Bakcend
  - Node.JS
  - JPA + Hibernate
  - EJB + CDI
- Frontend
  - HTML5/CSS/AJAX
  - AngularJS + IONIC
- Rendszerek rendszere SoS
- DevOps

Alkalmazás rétegelés

- Az alkalmazás kompnensek feladatának elválasztása
  - Megjelenítés réteg
    - A kérés/válasz objektuomkra koncentrál
    - Megoldja a modell alapján a GUI renderelést
    - A validáció atomi, csak az egyes adatokat magukban kezeli
    - A más rétegek által dobott kivételeket kezeli
  - Perzisztencia réteg
    - A tárolókkal kommunikál
    - Lekérdezés nyelvet biztosít
    - ORM képesség
    - JDBC, Hibernate, iBATIS, JDO, Entity Beans, …
  - Domén réteg
    - Az üzleti objektumokat tárolja, ezek a többi rétegben is felhasználhatóak
    - Kezeli az üzleti objektumok közötti komplex relációat
    - Gazdag üzleti logika
    - ORM

Szolgáltatás réteg?



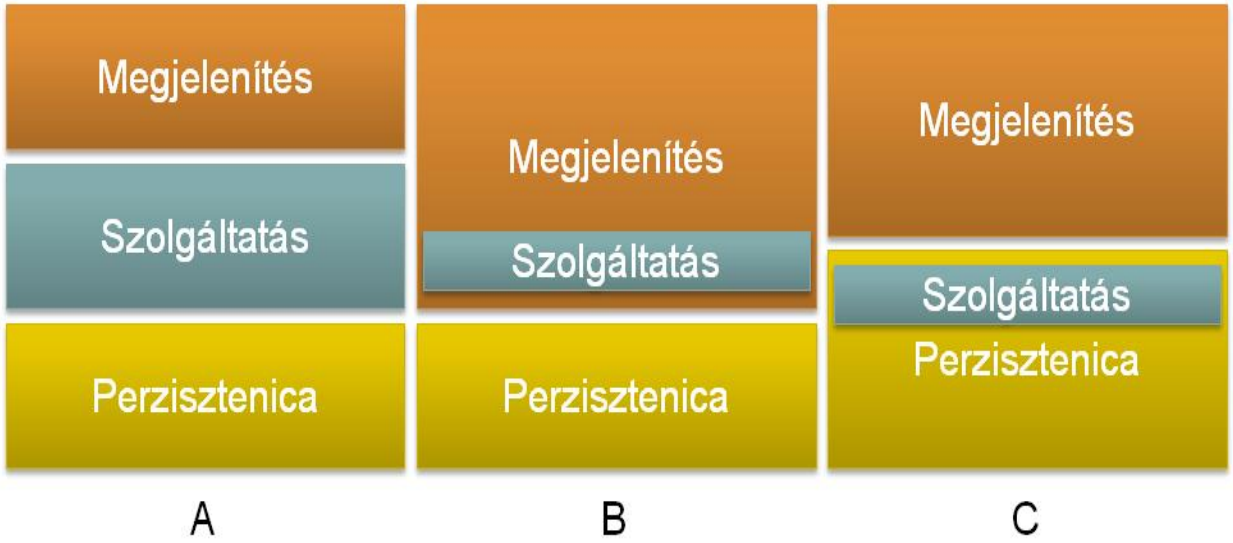
- Hogyan pozicionáljuk a lazán csatolt üzleti logikát?
- Mi a szolgáltatás logika?
- Hogyan kell a konténer szintű szolgáltatásokat megvalósítani?
- Hogyan kezeljük a POJO alapú alkalmazásokban a tranzakciókat?
- Hogyan kommunikáljunk a megjelenítési rétegből a perzisztencia rétegbe?
- Hogyan kapunk üzleti logikát megvalósító szolgáltatásokat?

Alkalmazás rétegek

- Szolgáltatás réteg
  - Egy átjáró amely segítségével az üzleti logikát biztosítjuk a külvilág számára
  - Kezeli a tároló szintű szolgáltatásokat (tranzakciók, biztonság, …)ß

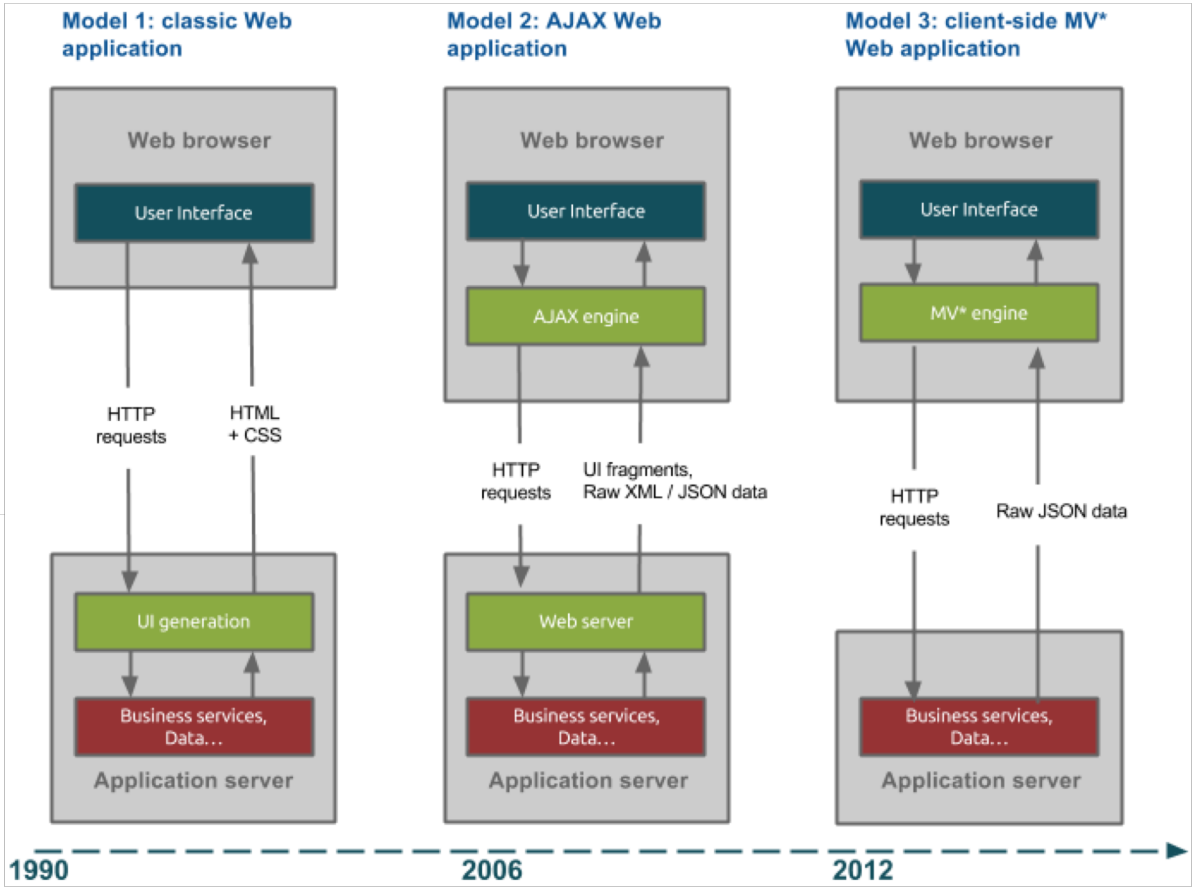
Alkalmazás architektúra

- 3 Rétegű Vs. MVC



Alkalmazás architektúrák

- Web architektúrák



Mikroszolgáltatások

1990s and earlier | **2000s** | **2010s**

Coupling

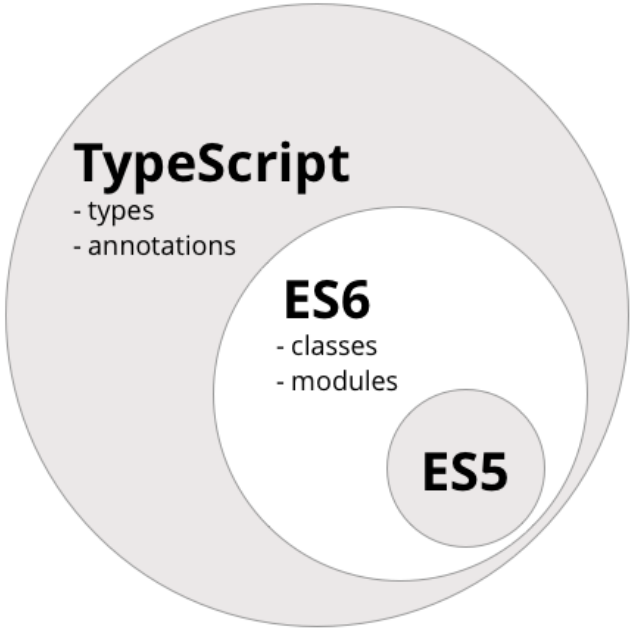
Pre-SOA (monolithic)
Tight coupling

Traditional SOA
Looser coupling

Microservices
Decoupled

2. Óra: TypeScript

- ES6, Typescript
- Környezet (GIT, GRUNT, LINT, BABEL)
- REPL
- Futtató környezet
- Változók, Konstansok, Adattípusok
- Objektumok
- Kollekciók
- Függvények
- Függvény kifejezések
- Szkópok
- Zárványok
- IIFE
- Aszinkron programozás (callback, promise)



3. óra: Elosztott rendszerek

- Internet skálájú rendszerek
- Nem funkcionális követelmények
- Rendelkezésreállás (SLA)
- Rendszer állapotok, MTTF, ...
- Rendszer elérhetetelenség okok
- Megoldások
- Redundancia
- Példák
- Skálázhatóság
- Felhő IAAS, PAAS
- CAP
- Nem CA,AP,CP
- Konzisztencia típusok
- Partícionálások
- PACELC

Valós nagy rendszerrel kapcsolatos tapasztalatok (Inktomi)

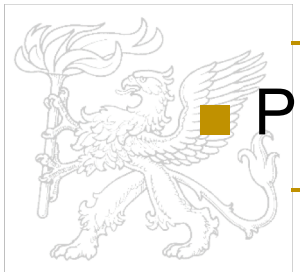
Összeomlások/Merevlemez hibák	Óránként
Adatbázis frissítések	Naponta
Szoftver frissítések	Hetente/Havonta
OS frissítések	Kétszer
Energetikai hibák	néhány
Hálózat hibák	11
Az egész infrastruktúra fizikai átköltöztetése	2

Skálázhatóság

- Adat tárolás/Feldolgozás
  - Vertikális (Scale up)
    - Fizikai korlátai vannak (processzorok száma, JVM szemétgyűjtő, …)
    - Közös memória mint kommunikációs médium
  - Horizontális (Scale out)
    - Elosztott rendszer (Leslie Lamport: „Elosztott rendszer az ahol egy addig teljesen ismeretlen számítógép hibája teszi használhatatlanná a gépünket”)
    - Elvileg korlátlan (elosztottságból fakadóan nem lehet ACID)
    - Távoli hozzáférés problematikája (érték szerint vs. referencia szerint)

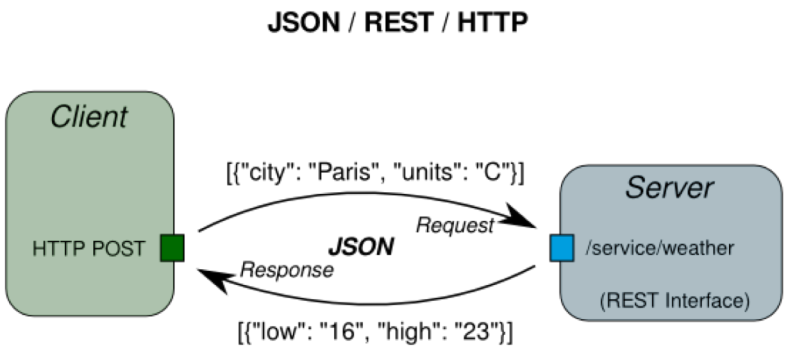
Határok – CAP tétel (Brewer tétele)

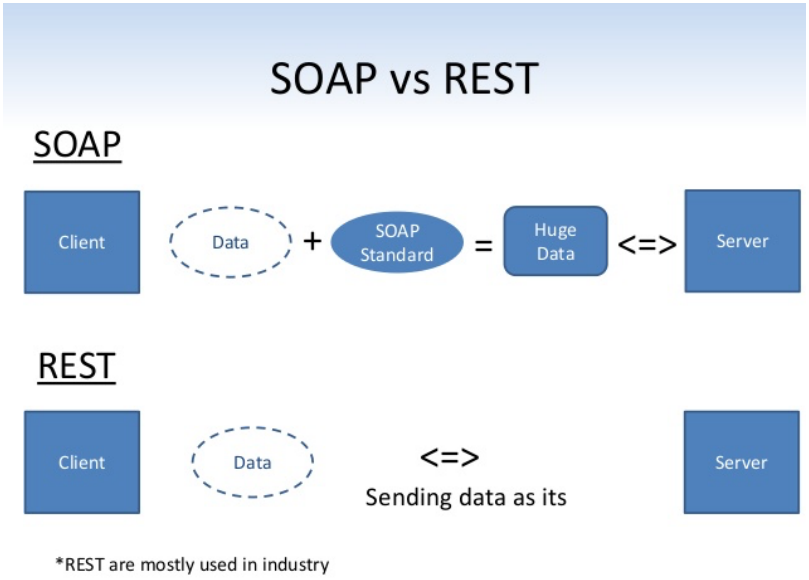
- Internet méretű elosztott rendszerek
  - C – Konzisztencia (Minden csomópont ugyanazt az adatot látja adott időben)
    - A – Atomi
    - C - Konzisztens
    - I - Elkülönítés
    - D - Tartósság
  - A - Rendelkezésre állás (csomópontok kiesése nem akadályozza meg a maradókat a működéstől)
    - X % időben elérhető
  - P - Partíció tűrés (tetszőleges üzenetvesztést tolerál)
    - Gilber és Lynch definíciója szerint „*A teljes rendszer (hálózat) hibáján kívül semmilyen hiba kombináció sem okozhatja azt, hogy a rendszer hibásan válaszol*”.
- Ezek közül csak kettő teljesülhet



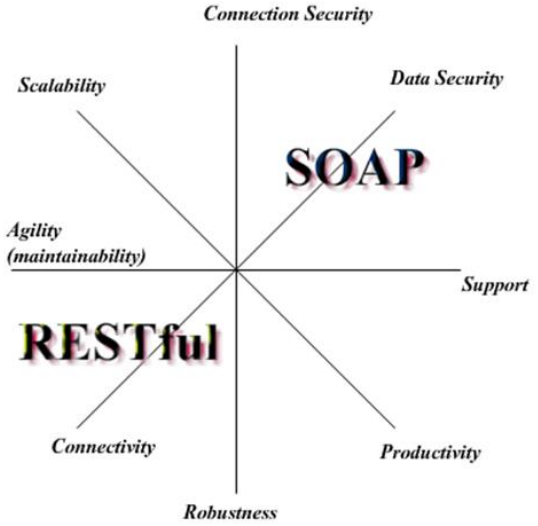
4. Óra: Web Services/REST

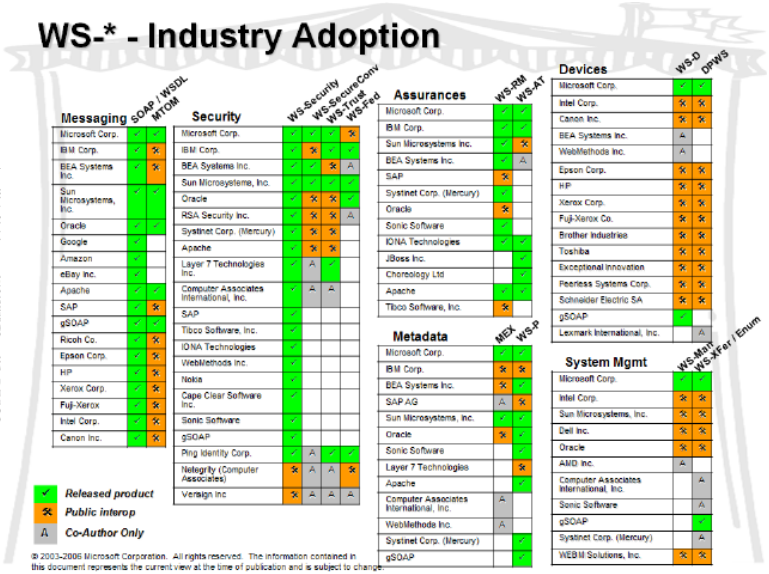
- HTTP
- RMI...
- REST vs WS
- REST
- WS
- SOAP
- WSDL
- WS profilok

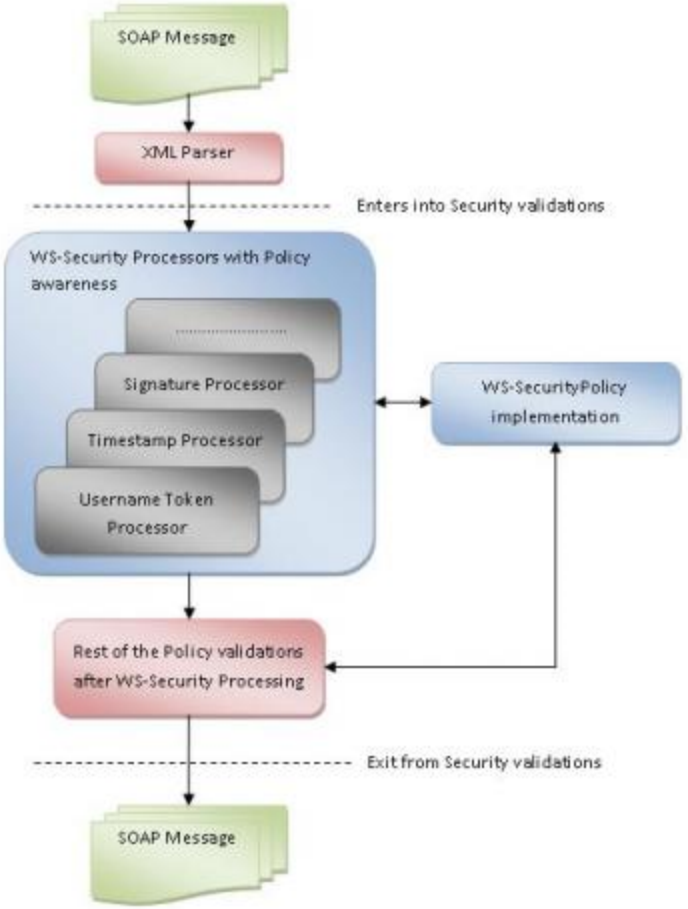




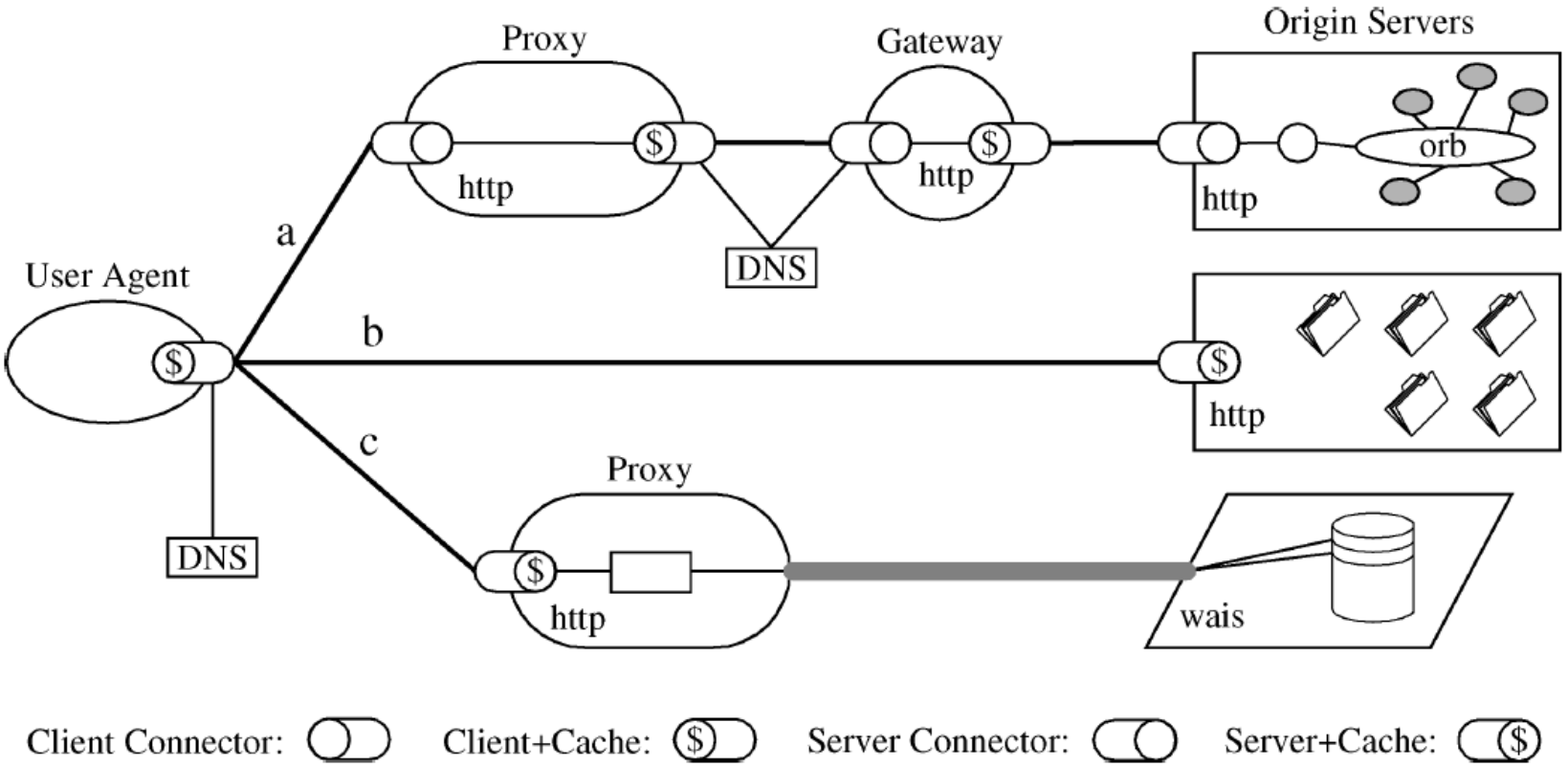
Rest/Web Szolgáltatások





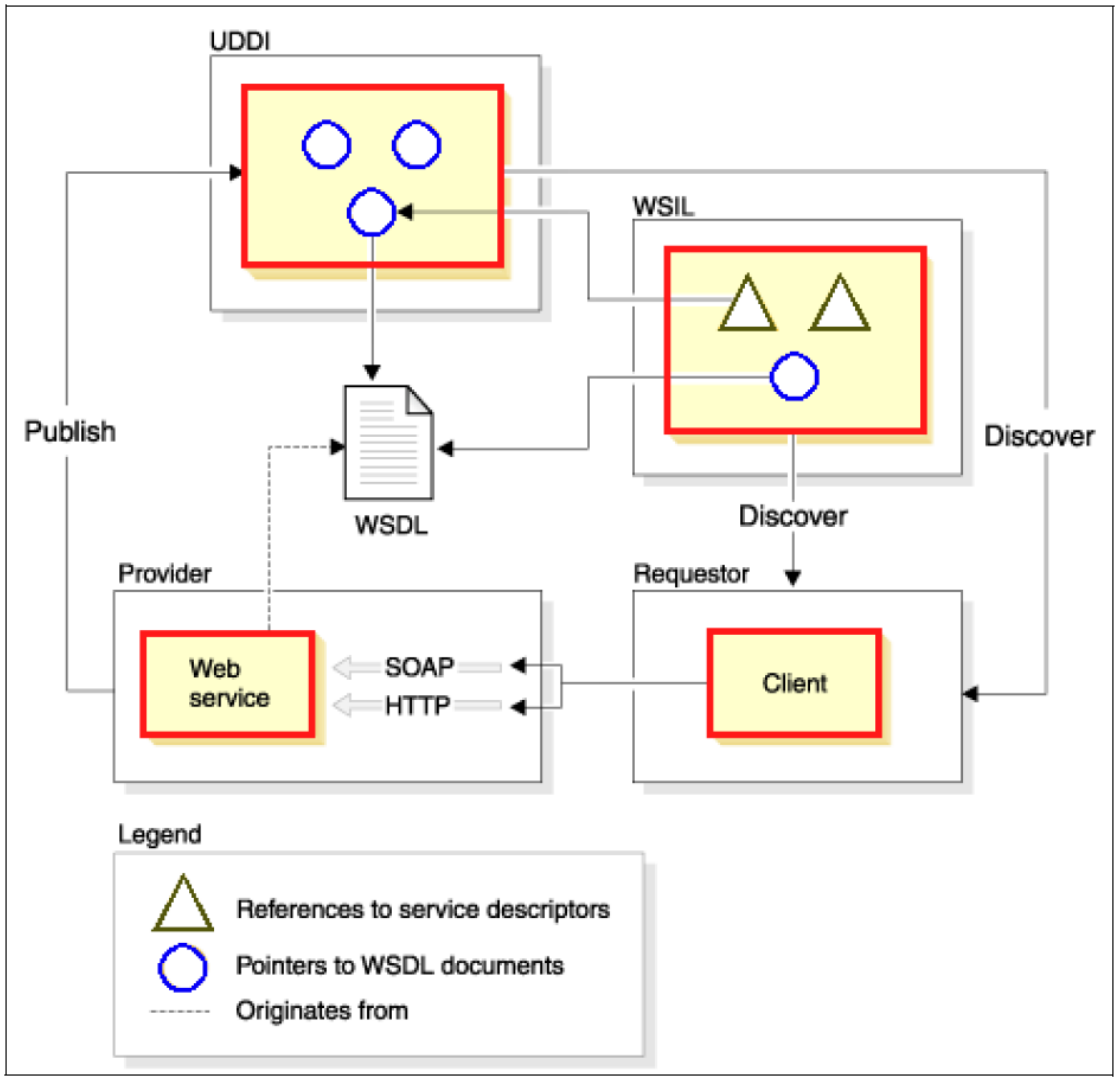


REST architektúra



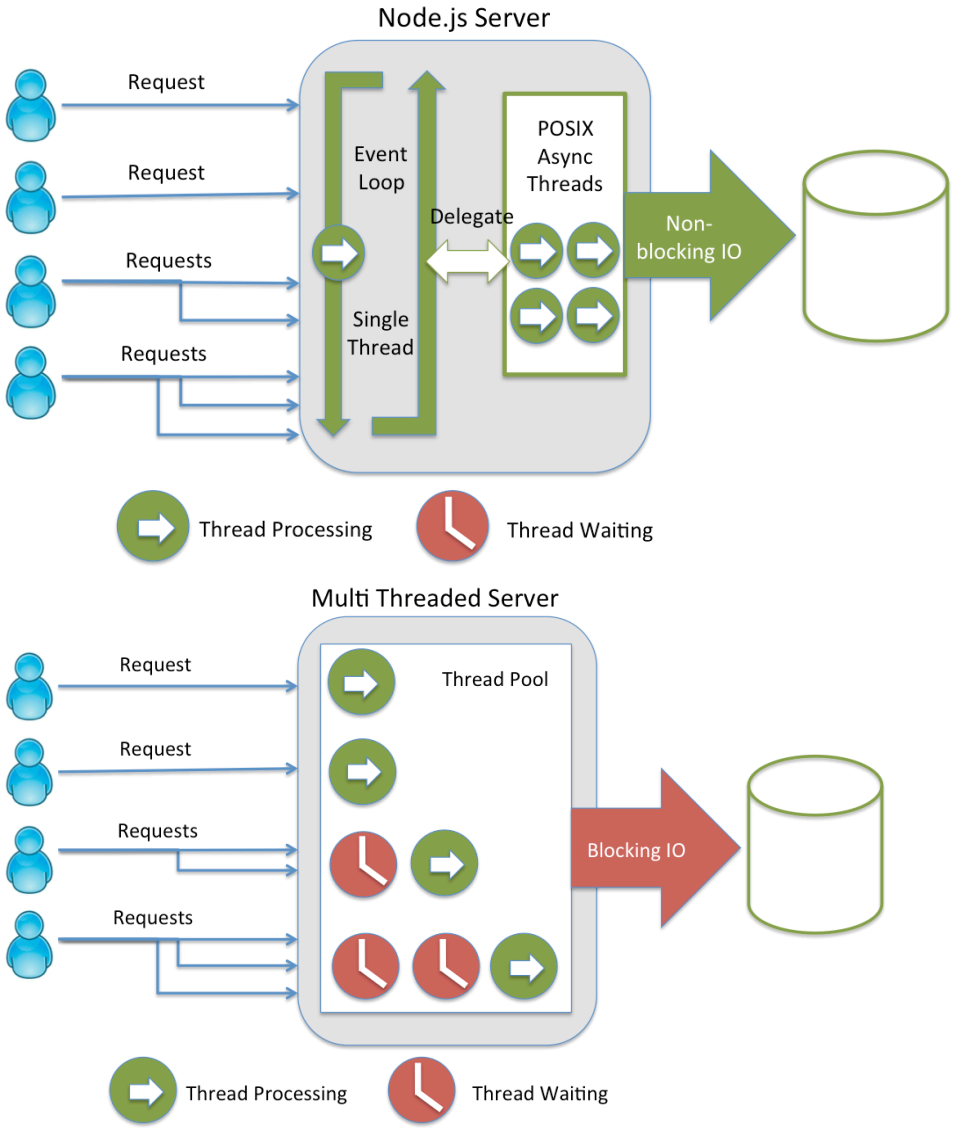
A SOA fő elemei

- XML
- SOAP
- WSDL
- WSIL
- UDDI



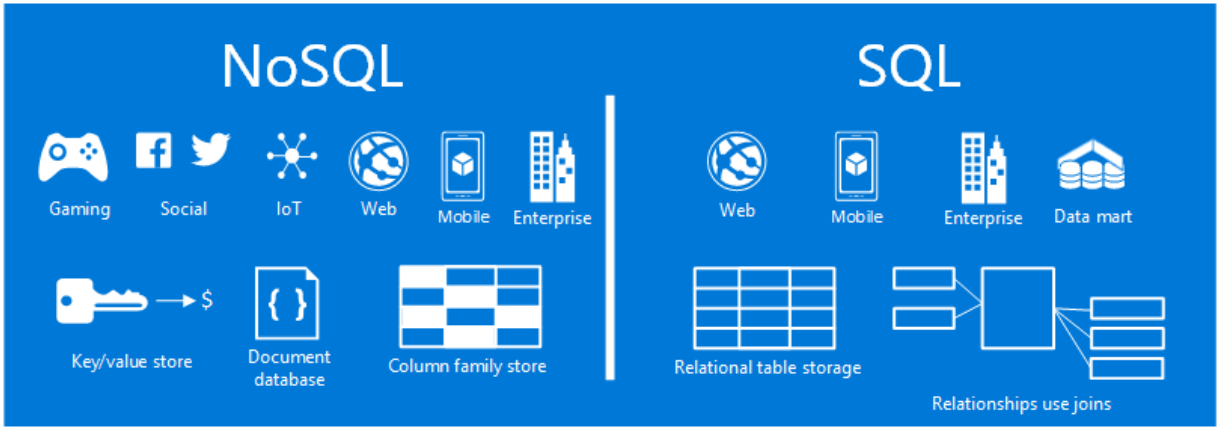
5. óra: Háttér logika Node.JS

- Web backend
- Little törvénye, skálázhatóság
- 3. generációs háttér
- Esemény hurok
- Szálak vs Esemény vezérelt
- NodeJS
- Eseményvezérelt programozás
- Blokkoló/Nem blokkoló
- Modulok, változók, globális objektumok
- Alap modulok
- NPM
- REPL
- Express, Routing
- RESTFull API



6. óra: Adatkezelés: NoSQL

- Adattípusok
- Történet
- Skálázási megoldások
- Replikáció
- Konzisztencia - Commit protokollok
- RDBMS - sharding
- Amdahl törvénye
- NoSQL
- Dynamo - Bigtable
- CAP újra
- Elég jó konzisztencia
- BASE
- NoSQL típusok
- Dokumentum tár
- Gráf adatbázis
- Név-érték pár
- Oszlopos adatbázisok

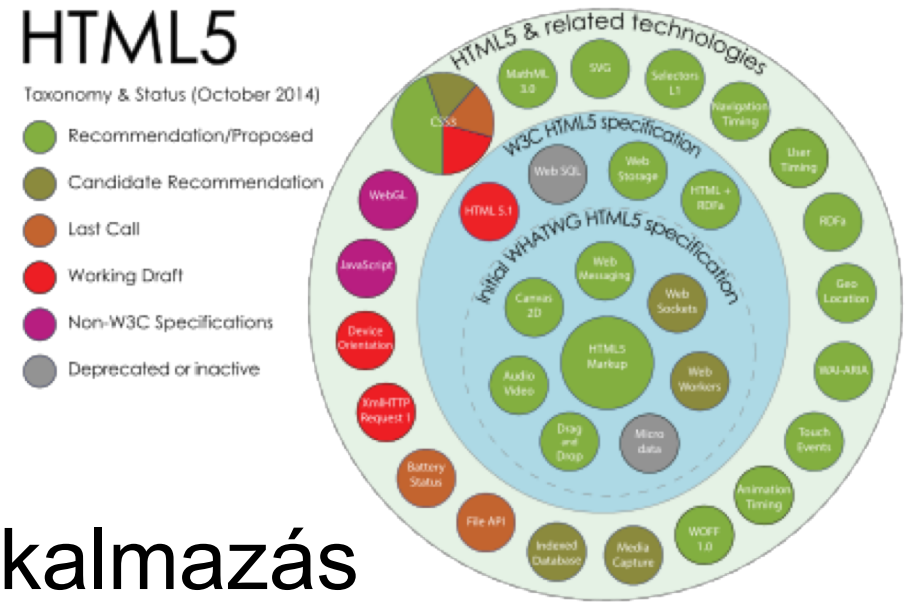


7. óra: MongoDB

- Motiváció
- Adat modell
- JSON, BSON
- Gyűjtemények
- CRUD
- Korlátok
- Séma
- Indexelés
- Aggregáció, csövek, map-reduce
- Replikáció, sharding

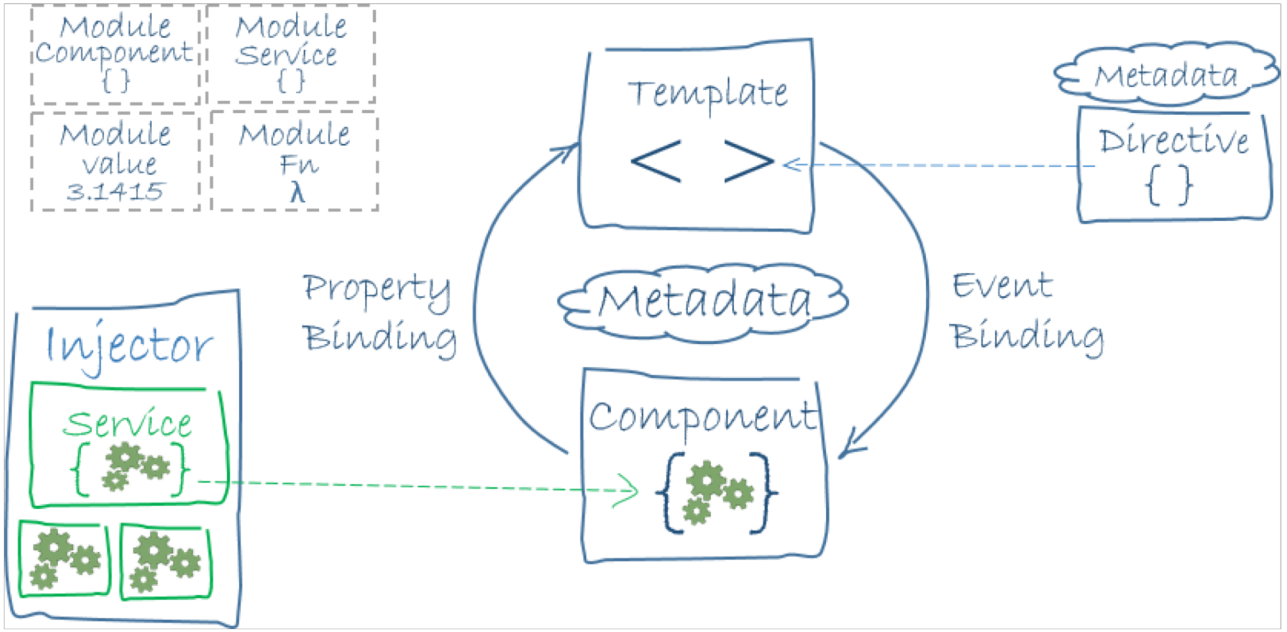
8. óra: GUI

- Trendek: RIA, SPA, ...
- HTML5
- AJAX
- DOM
- CSS
- Reszponzív web alkalmazás
- Bootstrap



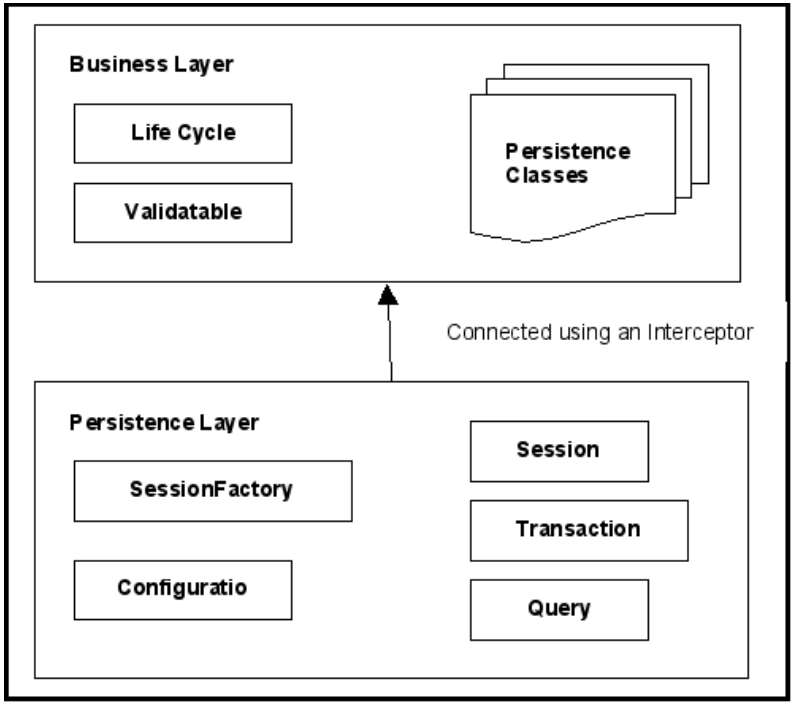
9. óra: Angular2 + IONIC2

- SPA
- Angular megközelítés
- Modulok
- Komponensek
- Sablonok
- Metainformációk
- Dekorációk
- Adatkötés
- Direktívák
- Szolgáltatások
- Injketálás
- Alap dirketíva készlet
- Változás detektálás
- Forgalomirányítás
- HTTP
- Asszink programozás
- Reaktív programpozás, RxJS
- IONIC



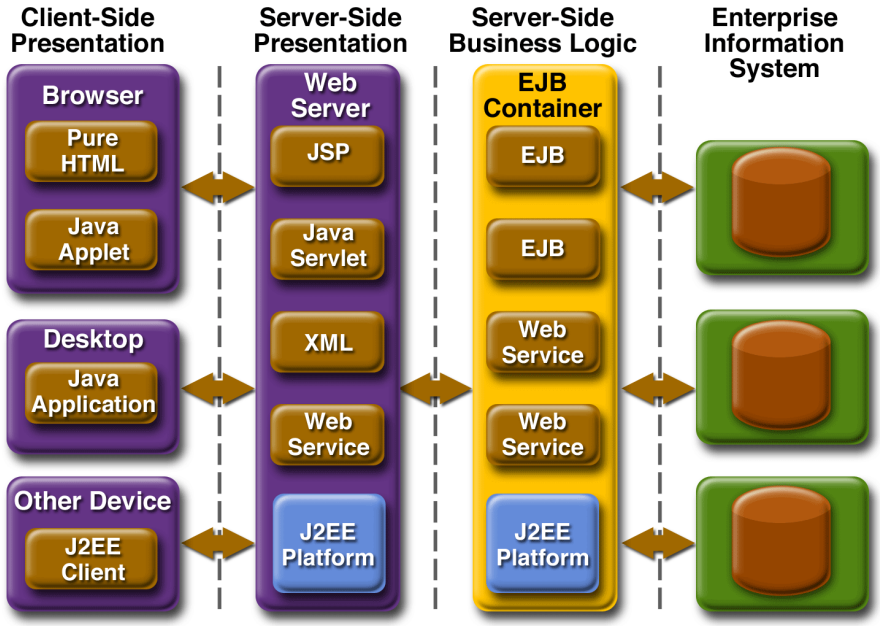
10. óra Adatkezelés

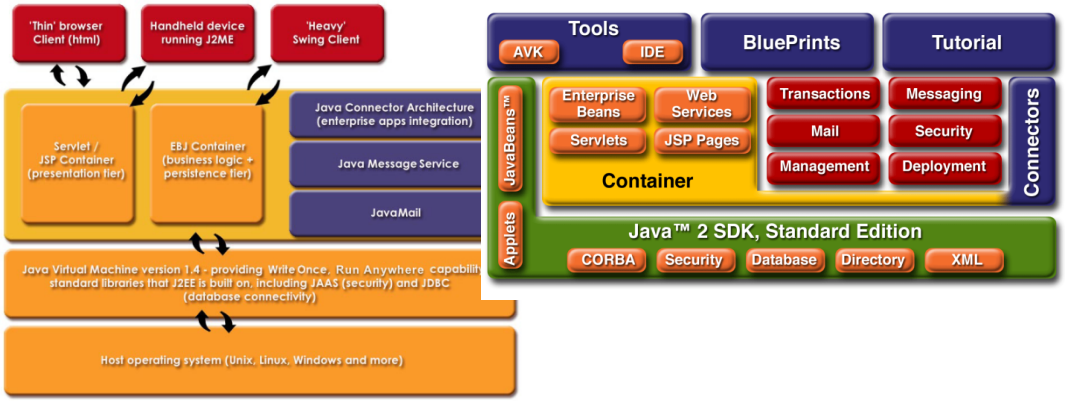
- Perzisztencia
- Object serialization API
- ORM
- Hibernate
  - Architecture
  - Collections, Polymorfisc, etc.
  - Data manipulation
  - Optimilisation (fetching and caching)

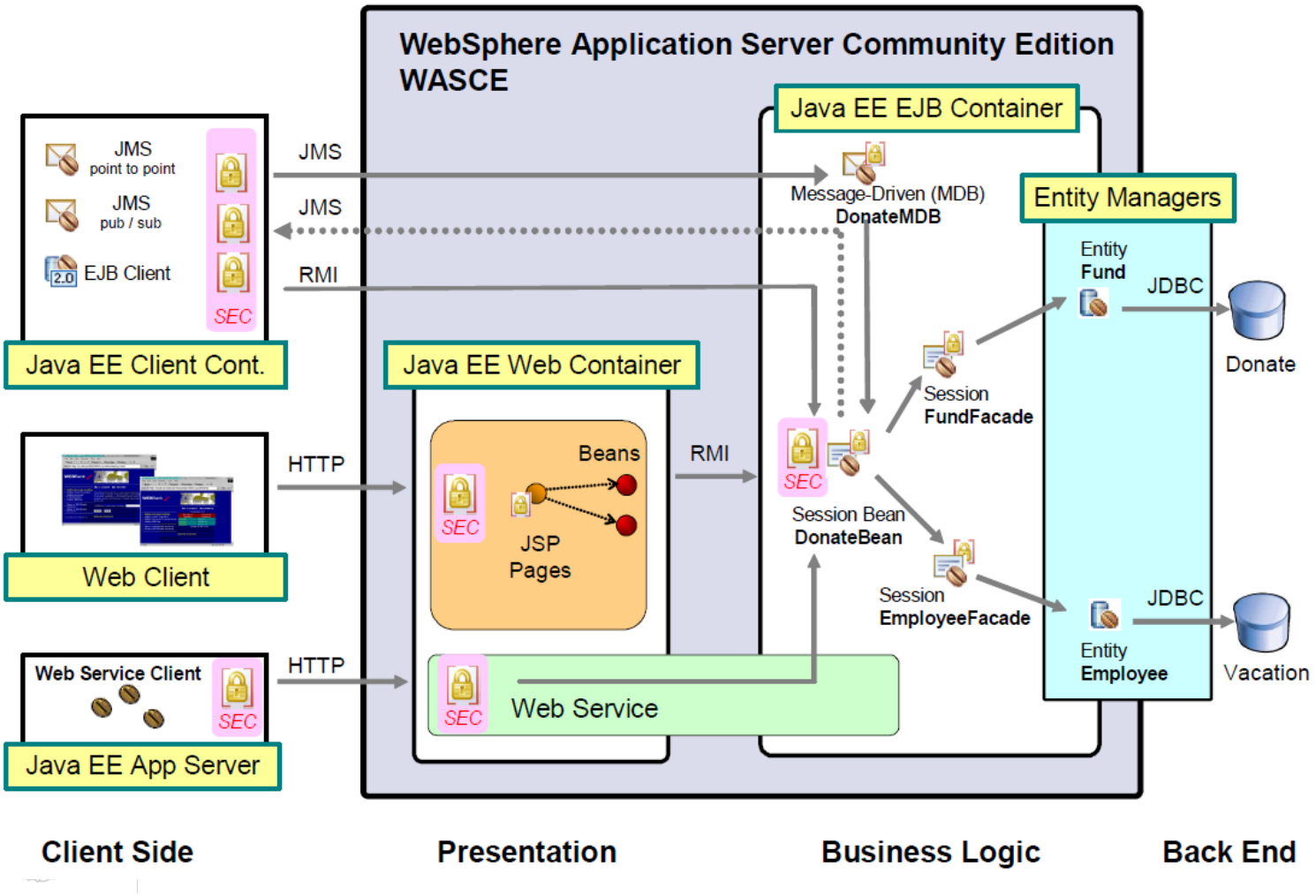


11. Óra Háttér

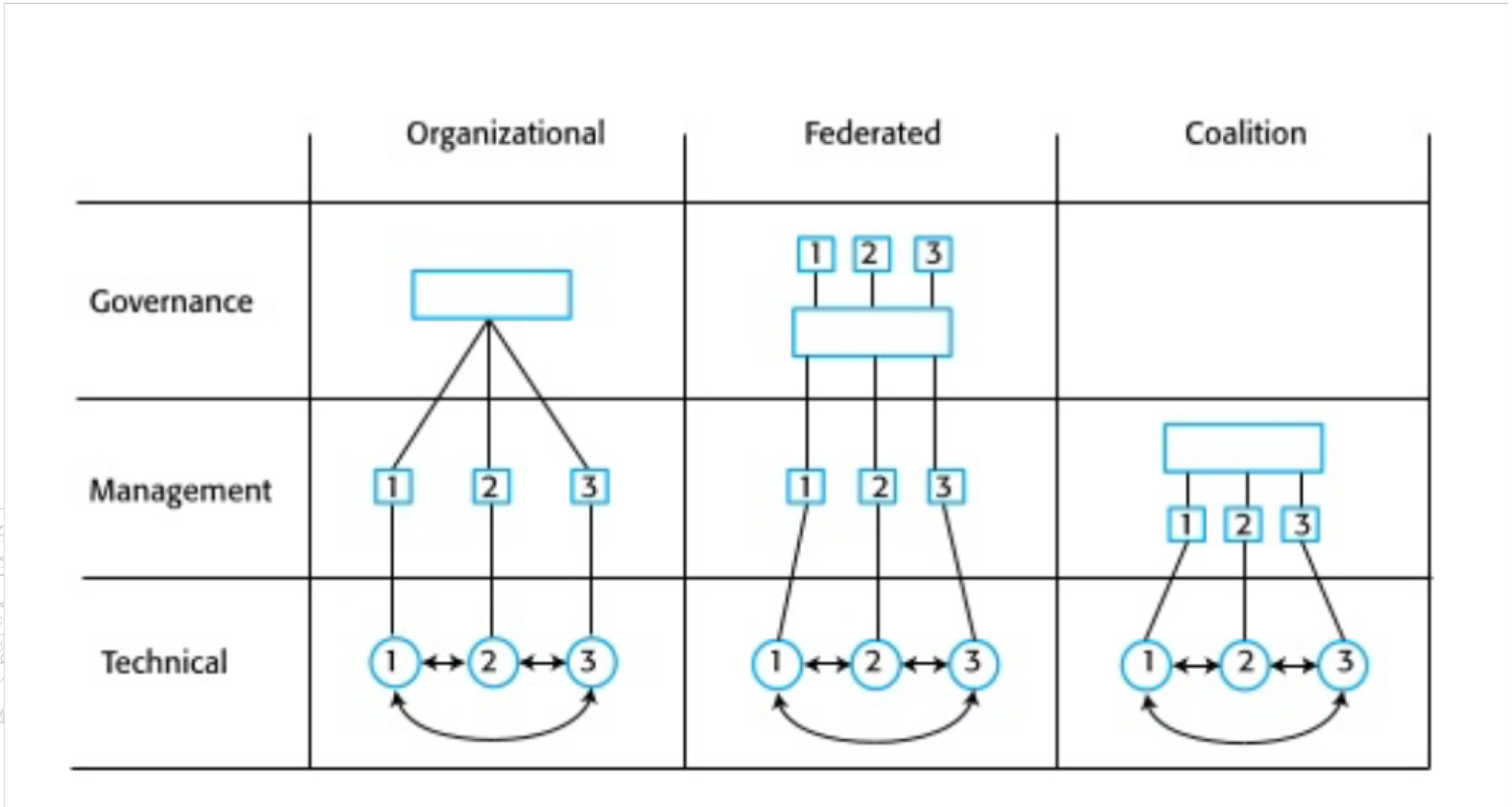
- Java Enterprise Edition
  - EJB specification(3.0)
  - EJB components
- Injection
- Scope
- Transaction
- EJB
  - Session Bean
    - Stateless
    - Statefull
  - Entity Bean
    - BMP
    - CMP
  - Message Driven Bean
    - Durable
    - Non Durable



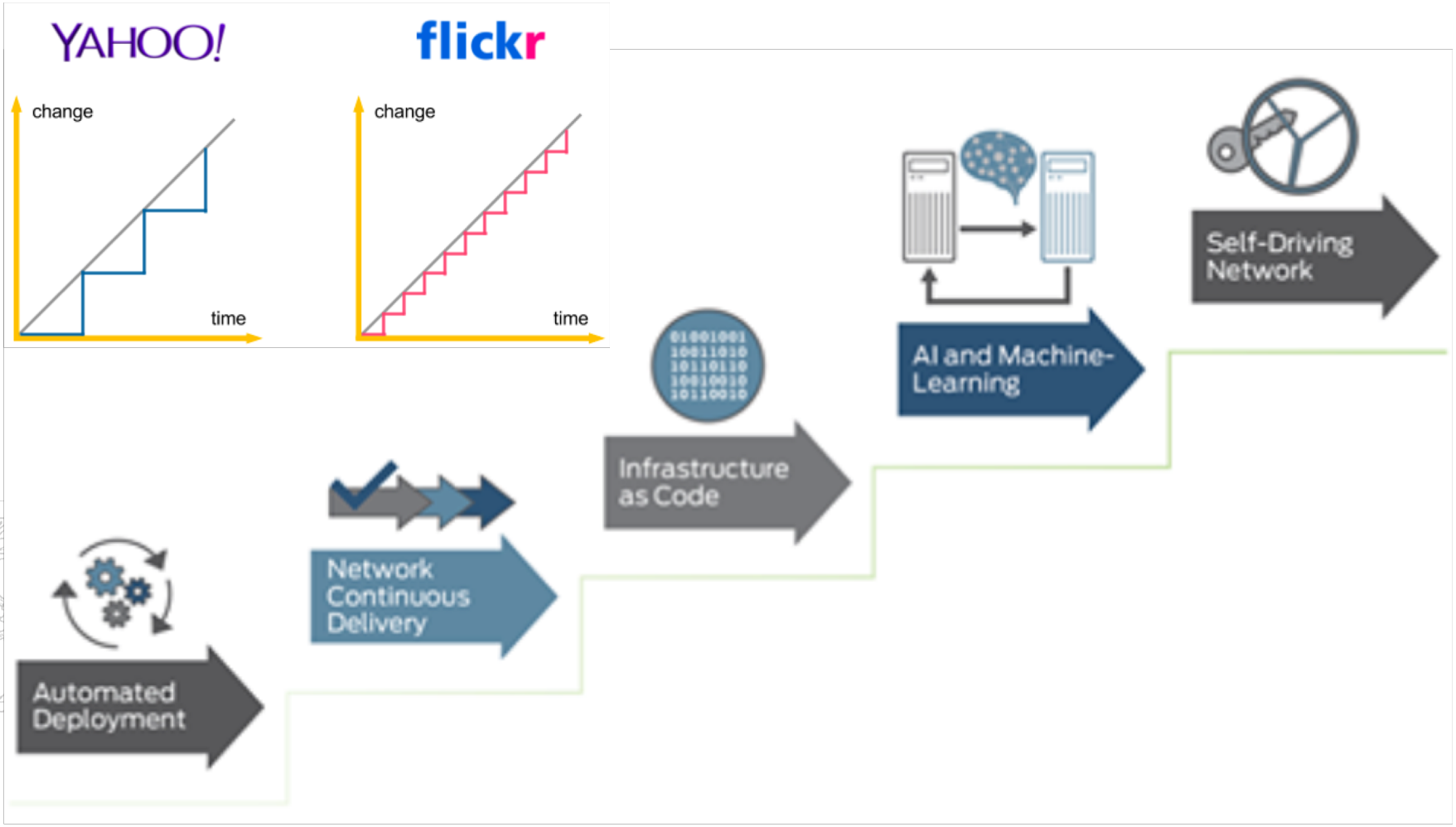



WebSphere Application Server Community Edition
WASCE
Java EE EJB Container
Entity Managers
JMS
point to point
JMS
pub / sub
EJB Client
SEC
Java EE Client Cont.
JMS
JMS
RMI
Message-Driven (MDB)
DonateMDB
Entity
Fund
JDBC
Donate
Java EE Web Container
Beans
JSP
Pages
RMI
Session
FundFacade
Session Bean
DonateBean
Session
EmployeeFacade
Entity
Employee
JDBC
Vacation
HTTP
Web Client
Web Service Client
HTTP
Java EE App Server
Web Service
Client Side
Presentation
Business Logic
Back End

SoS – System of system engineering



DevOps


YAHOO!
change
time
flickr
change
time
Automated Deployment
Network Continuous Delivery
Infrastructure as Code
AI and Machine-Learning
Self-Driving Network

Összefoglaló

- Motiváció – nem funkcionális/rendszer követelmények
  - Rendszer granularitás paradigmák
  - Skálázhatóság
- Internet alapú rendszerek
  - Korlátai
  - Lehetőségeink
- Elosztott rendszer architektúrák
- Átszövődő vonatkozások
- Köztesréteg
  - Típusai
  - Megvalósítása
  - Metrikái
  - Példák