



Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής & Τηλεπικοινωνιών

Πληροφορικά Συστήματα

7ο Εξάμηνο 2020-21

Αρχιτεκτονικά πρότυπα

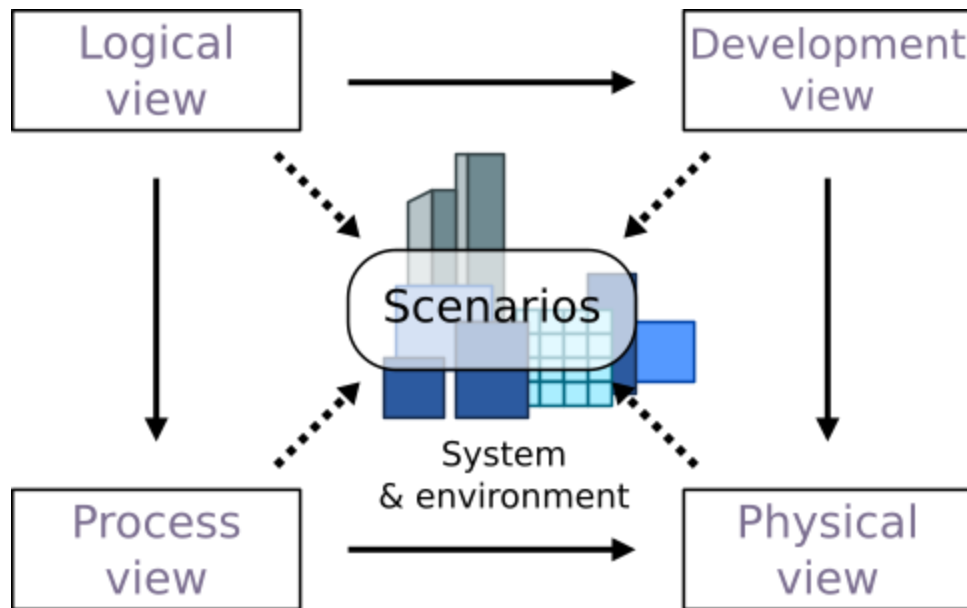
Δρ. Κώστας Σαΐδης (saiko@di.uoa.gr)

Αρχιτεκτονική λογισμικού (γενικά)

Περί τίνος πρόκειται

Η λήψη των θεμελιωδών δομικών και σχεδιαστικών αποφάσεων για το λογισμικό που είναι ακριβό (σε κόπο, χρόνο, χρήμα) να αλλάξουν άπαξ και υλοποιηθούν.

Πολλές αρχιτεκτονικές οπτικές (4+1)



By mpan - Based on File:4+1 Architectural View Model.jpg by User:Mdd, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=50144028>

Ειδικότερα

- Logical view: έμφαση στη λειτουργικότητα του συστήματος σε υψηλό επίπεδο
- Physical view: έμφαση στην τοπολογία και διασύνδεση των συστατικών του σε φυσικό επίπεδο (deployment)
- Development view: έμφαση στην οπτική του προγραμματιστή
- Process view: έμφαση στη δυναμική συμπεριφορά του συστήματος κατά την εκτέλεσή του (απόδοση, κλιμάκωση, κτλ.)
- Scenarios - Use case view: έμφαση στη χρηστική πλευρά του συστήματος και στους σχετικούς ελέγχους αποδοχής

Επιχειρησιακή Αρχιτεκτονική (Enterprise Architecture)

- Ολιστική οπτική των αρχιτεκτονικών ζητημάτων που εμπλέκονται στον Οργανισμό, στα Δεδομένα του, στις Διαδικασίες του και στην ΤΠΕ
- Διατρέχει το σύνολο των ΠΣ του Οργανισμού
- Θα επανέλθουμε σε επόμενη διάλεξη

Αρχιτεκτονικά πρότυπα (architectural patterns)

Γενικές κι επαναχρησιμοποιήσιμες λύσεις σε κοινά προβλήματα αρχιτεκτονικής.

Θα κουβεντιάσουμε σήμερα (αλφαβητικά)

- Client-Server
- Component-based
- Layered / N-tier
- Master-slave/Master-replica
- Message-driven/Publish-subscribe
- Microservices

- Peer-to-peer (P2P)
- Pipeline / Pipe-filter
- Representation State Transfer (REST)
- Service-oriented
- Share-nothing architecture
- World Wide Web

Cloud computing

Σε επόμενη διάλεξη

Πριν ξεκινήσουμε

Βασικές έννοιες κατανεμημένων συστημάτων

Κατανεμημένα συστήματα

Ένα κατανεμημένο σύστημα αποτελείται από ξεχωριστά συστατικά που:

- Λειτουργούν σε ένα δίκτυο υπολογιστών.
- Επικοινωνούν μεταξύ τους μέσω ανταλλαγής μηνυμάτων.
- Αλληλοεπιδρούν για την επίτευξη ενός κοινού στόχου.

Βασικά χαρακτηριστικά ενός κατανεμημένου συστήματος

- Συνέπεια δεδομένων (Consistency)
- Διαθεσιμότητα συστήματος (Availability)
- Αστοχία δικτύου (Network partition)
- Καθυστέρηση αίτησης/απόκρισης (Latency)
- Αιτήσεις ανά μονάδα χρόνου (Throughput)
- Κλιμάκωση (Scalability)

Consistency (C)

- Η συνέπεια των δεδομένων.
- Ελάχιστο ζητούμενο: κάθε ανάγνωση (read) λαμβάνει την πιο πρόσφατη ενημέρωση (write) ή το σχετικό σφάλμα.

Availability (A)

- Η διαθεσιμότητα της εφαρμογής.
- Ζητούμενο: κάθε αίτηση (request) να λαμβάνει μια απάντηση (μη λάθους).
- Χωρίς να προσφέρονται πάντα εγγυήσεις ότι η απάντηση περιέχει την πιο πρόσφατη ενημέρωση (write).
- Η υψηλή διαθεσιμότητα απαιτεί αντιγραφές (replication).

Network Partition (P)

- Αστοχία δικτύου.
- Επιμερισμός του συστήματος σε "αποσυδεδεμένες" νησίδες.
- Παράδειγμα: απώλεια σύνδεσης με τη βάση δεδομένων.

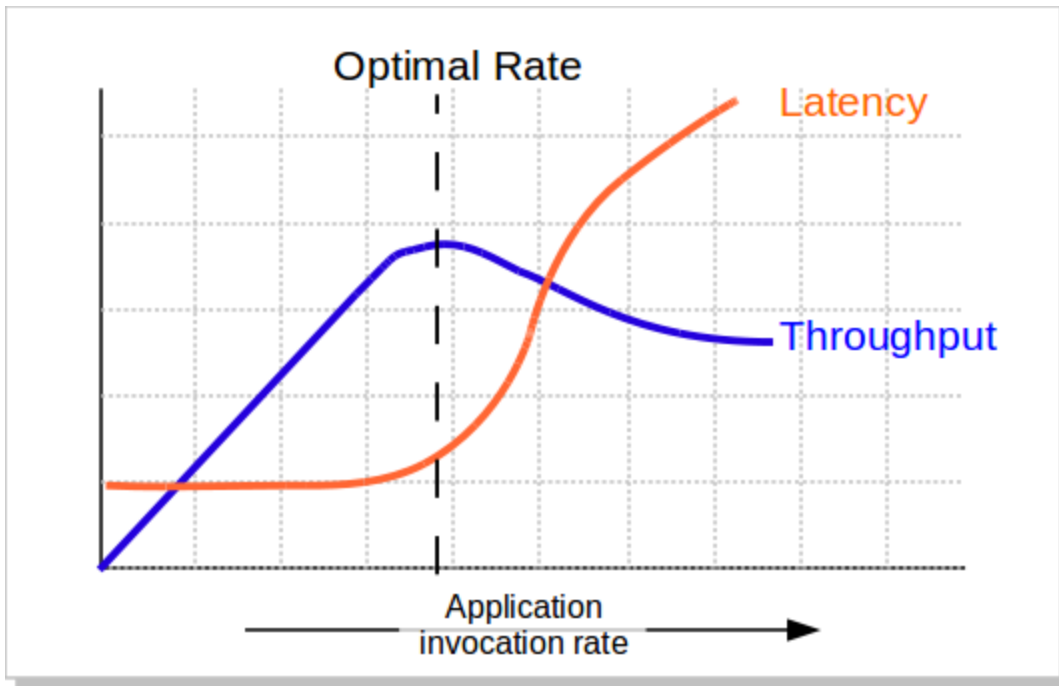
Latency

- Η καθυστέρηση στην απόκριση του συστήματος.
- Ζητούμενο: να ελαχιστοποιηθεί ο χρόνος που απαιτείται για την ικανοποίηση μιας αίτησης.

Throughput

- Το πλήθος των αιτήσεων που ικανοποιούνται από το σύστημα ανά χρονική στιγμή.
- Ζητούμενο: να μεγιστοποιηθεί το πλήθος των αιτήσεων που μπορούν να ικανοποιηθούν ανά χρονική στιγμή.

Latency vs Throughput



docs.voltdb.com

Scalability

Η δυνατότητα ενός συστήματος να επαυξηθεί για να διαχειριστεί αυξημένο φόρτο.

Ειδικότερα

Αν η αύξηση της απόδοσης του συστήματος είναι αναλογική της αύξησης σε υπολογιστικούς πόρους (προσθήκη υλικού), τότε το σύστημα κλιμακώνεται (scales).

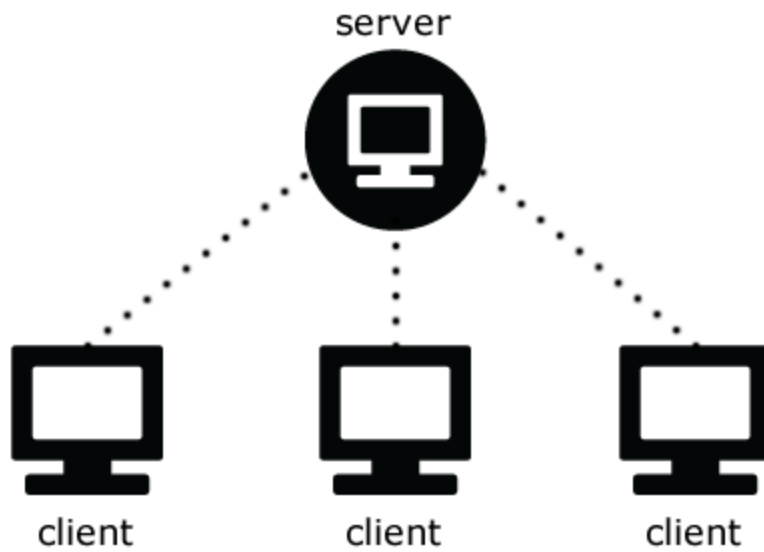
Οριζόντια και κάθετη κλιμάκωση

- Οριζόντια (scale out/in): αύξηση/μείωση των κόμβων.
- Κάθετη (scale up/down): αύξηση/μείωση των πόρων ενός κόμβου.

Επιστροφή

Στα αρχιτεκτονικά πρότυπα

Client-Server



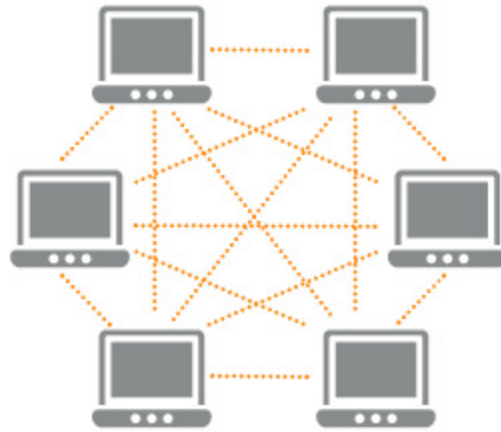
Χαρακτηριστικά

- Server-based
- N clients, 1 server
- Που επικοινωνούν μεταξύ τους με ένα συγκεκριμένο πρωτόκολλο για να υλοποιήσουν μια συγκεκριμένη "εφαρμογή"
- Παραδείγματα: WWW, IMAP, POP3, FTP, SSH, κ.ά

Peer-to-peer (P2P)



Server-Based

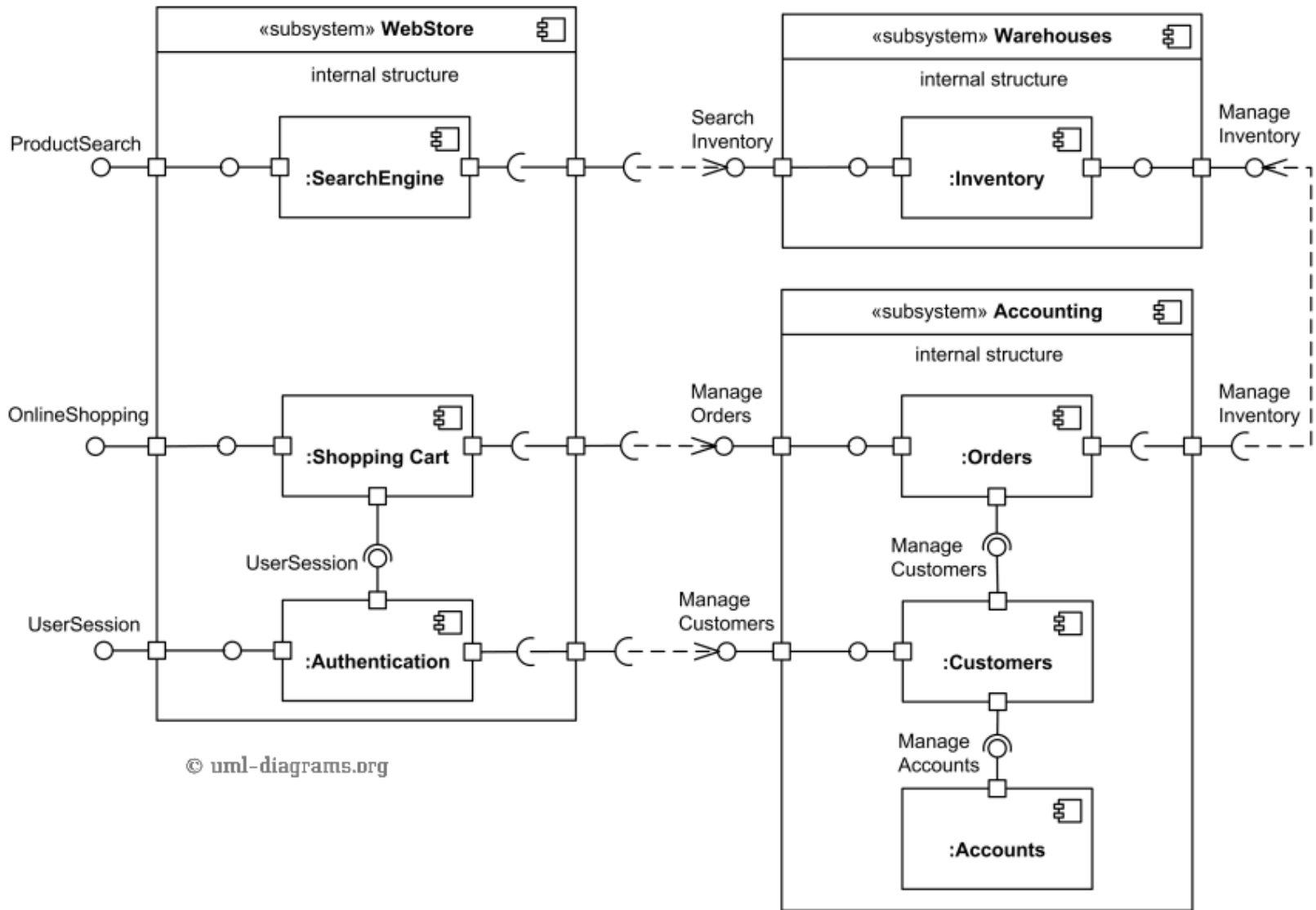


P2P

Χαρακτηριστικά

- Δίκτυο ομότιμων κόμβων
- Κάθε κόμβος είναι και client και server
- Οι κόμβοι επικοινωνούν μεταξύ τους με ένα συγκεκριμένο πρωτόκολλο για να υλοποιήσουν μια συγκεκριμένη "εφαρμογή"
- Παραδείγματα: File-sharing networks, Blockchain, Cryptocurrencies, κ.ά

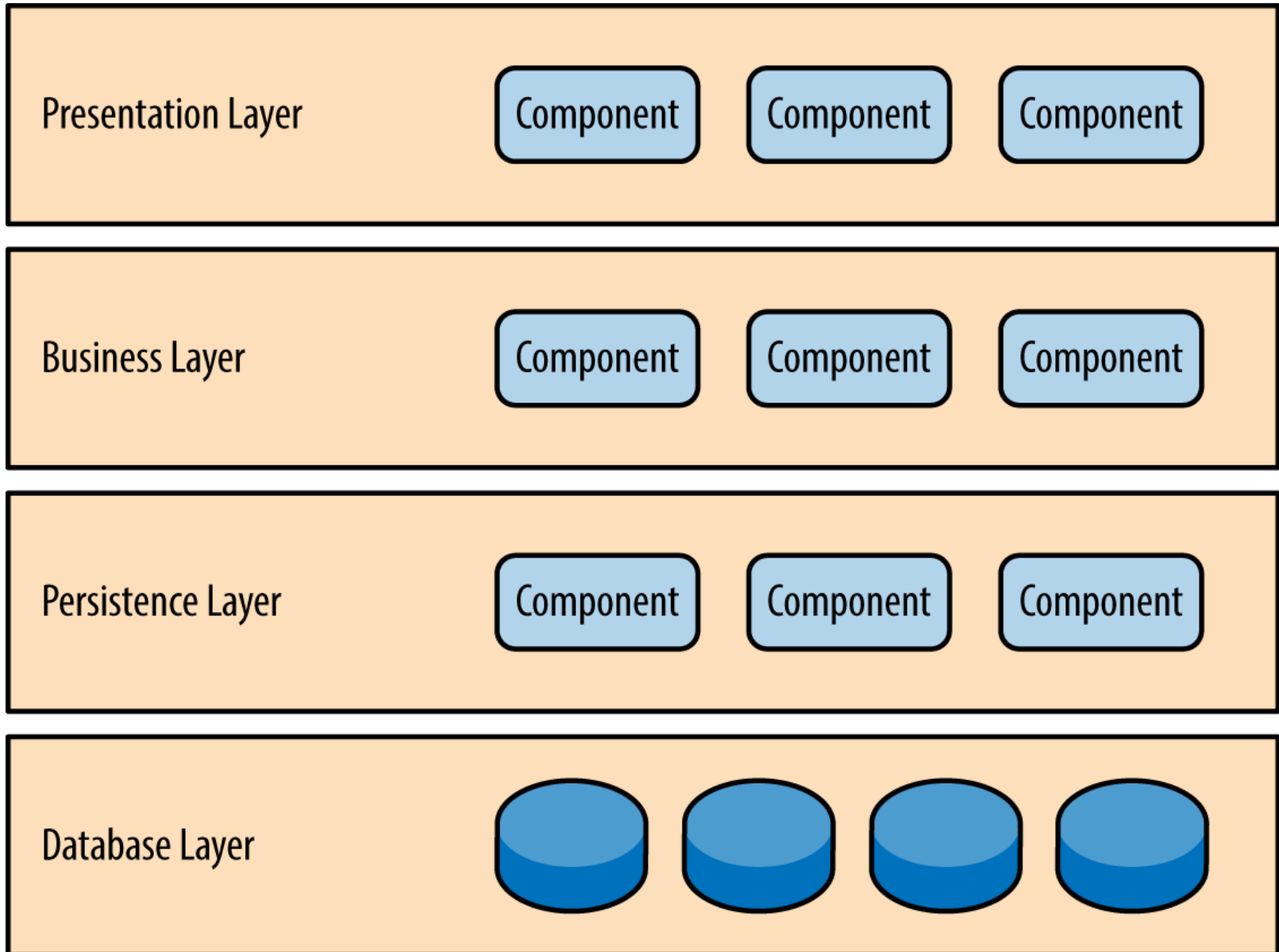
Component-based



Χαρακτηριστικά

- Σχεδιασμός και αλληλεπίδραση των συστατικών του λογισμικού μέσω Interfaces
- Ένα component παρέχει/υλοποιεί ένα interface και απαιτεί την ύπαρξη/κάνει χρήση ενός άλλου
- Χαλαρή σύνδεση (loose coupling) και διαχωρισμός ενδιαφερόντων (separation of concerns)
- Application server: το λογισμικό που φιλοξενεί τα components

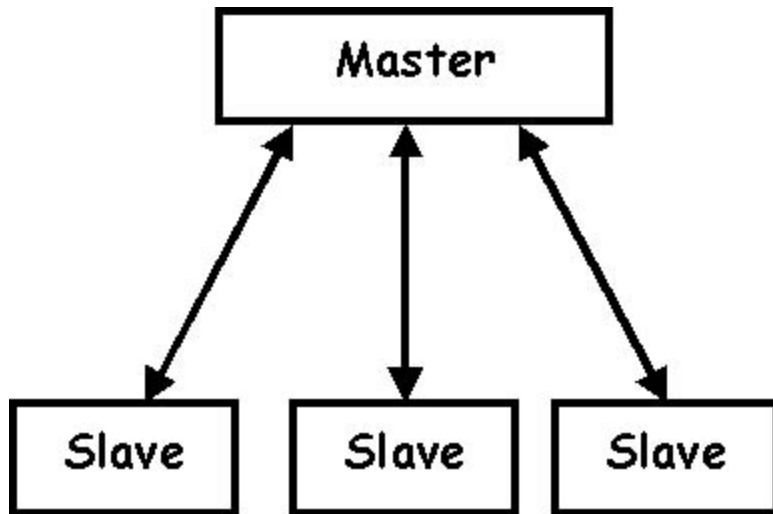
Layered/N-tier



Χαρακτηριστικά

- Server-based
- Λογική ή/και φυσική αρχιτεκτονική
- Ευρεία χρήση στις εφαρμογές διαδικτύου
- Frameworks: παρέχουν έτοιμα προς χρήση και παραμετροποίηση interfaces, components & layers

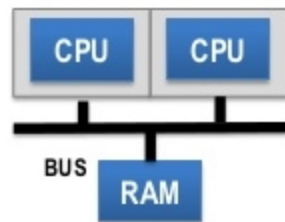
Master-Slave / Master-Replica



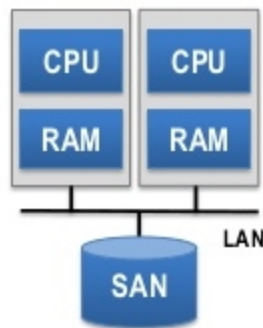
Χαρακτηριστικά

- N slaves, 1 master
- Master (authority), slaves (redundancy)
- Εφαρμογές: υψηλή διαθεσιμότητα, βελτίωση απόδοσης, επιμερισμός φόρτου, κ.ά
- Replication
 - Master copy of data, multiple replicas (slaves)
- Load balancing
 - Master dispatch logic, multiple "worker" nodes

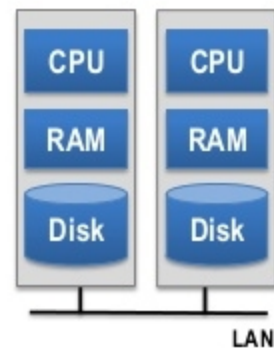
Share-Nothing Architecture



Shared RAM



Shared Disk



Shared Nothing

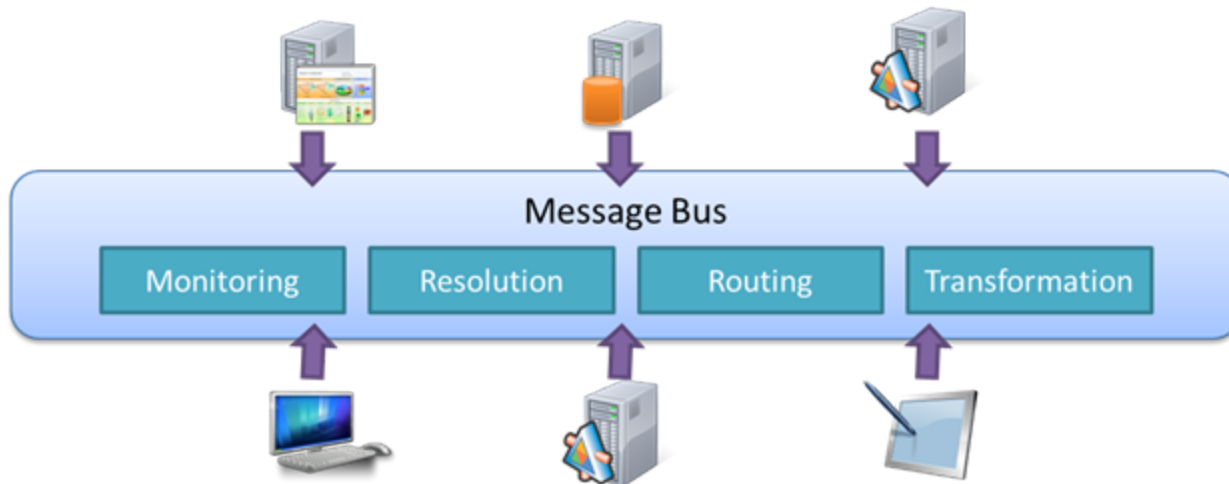
Χαρακτηριστικά

- Κάθε κόμβος είναι ανεξάρτητος και αυτοτελής.
- No single point of contention (δεν διαμοιράζονται πόροι, π.χ. μνήμη ή δίσκος).
- Sharding: οριζόντια επιμέρηση των δεδομένων.
- Οριζόντια κλιμάκωση (horizontal scalability) - απλή προσθήκη κόμβων.
- Η αρχιτεκτονική πολλών NoSQL συστημάτων.

Eventual Consistency

- **BASE Systems** (Basically Available, Soft state, Eventual consistency)
- Όταν πάψουν οι ενημερώσεις σε μια εγγραφή, τελικά (eventually) όλες οι αναγνώσεις της εγγραφής αυτής θα επιστρέψουν την πιο πρόσφατη ενημέρωση.
- Replica convergence (σύγκλιση αντιγράφων)
- PA/EL (Επιλέγουν αυξημένη διαθεσιμότητα & μείωση καθυστέρησης αντί για συνέπεια)

Message-driven/Publish-subscribe



Χαρακτηριστικά

- Χαλαρή σύνδεση (loose coupling) μεταξύ συστατικών/εφαρμογών
- Publisher (producer): αποστολή μηνυμάτων
- Subscriber (consumer): λήψη μηνυμάτων
- Topics (channels): "κλάσεις/θέματα" μηνυμάτων
- Message Bus (broker): διαχείριση/δρομολόγηση μηνυμάτων σύγχρονα ή ασύγχρονα, με εγγυήσεις αποστολής ή όχι, με χρήση ουρών, με φιλτράρισμα ή όχι κτλ.

Εφαρμογές

- Middleware ολοκλήρωσης ετερογενών συστημάτων
- Επίτευξη υψηλής απόδοσης και κλιμάκωσης σε κατανεμημένα συστήματα
- Μειονέκτημα: δύσκολη η αλλαγή της δομής των μηνυμάτων

Ο Παγκόσμιος Ιστός

Βασικά συστατικά

- Client/Server
- HTTP (Hyper Text Transfer Protocol)
- HTML (Hyper Text Markup Language)
- Unified Resource Identifiers, Locators and Names (URIs, URLs, URNs)

Μορφή ενός URL

```
scheme://[user:password@]host[:port]/path[?query][#fragment]
```

HTTP

- Μέθοδοι (HEAD, GET, PUT, POST, DELETE, κ.ά)
- Headers (Host, Accept, Cookie, κ.ά)
- Status codes (200, 404, 500, κ.ά)
- Εκδόσεις: 1.0, 1.1, 2.0

Παράδειγμα

Αίτηση

```
GET /index.html HTTP/1.1  
<line feed>
```

Απάντηση

```
HTTP/1.1 200 OK  
Date: Wed, 29 Mar 2017 14:38:00 GMT  
Server: Apache/1.3.27 (Unix) ...  
Last-Modified: Wed, 29 Mar 2017 01:16:05 GMT  
Accept-Ranges: bytes  
Content-Length: 6188  
Content-Type: text/html  
<html>  
<head>...</head><body>...</body>  
</html>
```

Pipeline / Pipe-filter



Χαρακτηριστικά

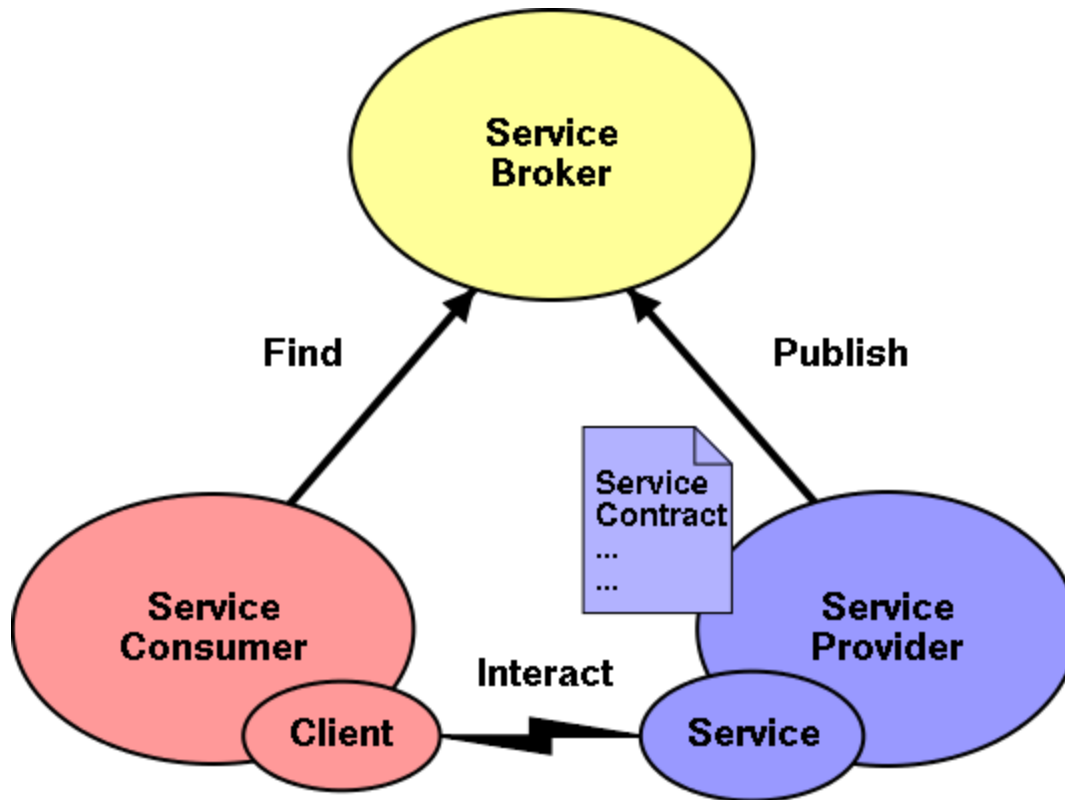
- Data streams, pipes and filters (data transformations)
- Συναρτησιακός προγραμματισμός
- Επαναχρησιμοποίηση, παραλληλισμός

Service-oriented architecture

Η κεντρική ιδέα

- Υπηρεσίες που επικοινωνούν μέσω ενός πρωτοκόλλου επικοινωνίας και είναι:
 - κατανεμημένες,
 - αυτοτελείς (separately maintained & deployed)
 - χαλάρα συνδεδεμένες (loosely-coupled)
 - ανεξάρτητες της τεχνολογίας υλοποίησης (technology-neutral)
 - ανεξάρτητες του κατασκευαστή (no vendor lock-in)
- Σύνθεση της εφαρμογής μέσω της ολοκλήρωσης (integration) των υπηρεσιών.

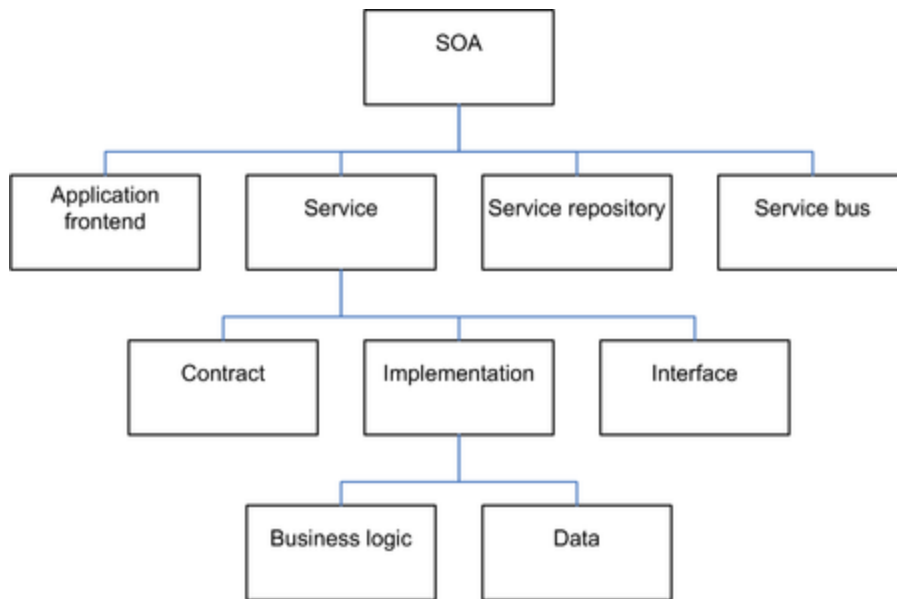
Ρόλοι



- Service consumer
- Service producer
- Service broker

SOA Αρχές

- Service contract
- Metadata
- Composition
- Autonomy
- Discovery
- Reusability



Υλοποίηση

- Web Services (SOAP, WSDL, UDDI)
- Remote-procedure call (RPC)
- Message-driven middleware
- RESTful APIs
- κ.ά

Ζητήματα

- Stateful vs Stateless
- Απόδοση
- Πολυπλοκότητα
- Έλεγχος και επαλήθευση (testing)

SOA & REST

- Ενδεχομένως ο απλούστερος τρόπος υλοποίησης μιας SOA
- Ένα σύνολο από RESTful HTTP end-points που ανταλλάσσουν δεδομένα σε μορφή JSON

Η Αρχιτεκτονική Representational State Transfer (REST)

RESTful APIs

- Υπάρχουν παντού (Web, Microservices, IoT)

Τι είναι το REST

- Ένα αρχιτεκτονικό στυλ για τη λειτουργία του Παγκόσμιου Ιστού (ή, γενικά, κατανεμημένων συστημάτων)
- Οριοθετεί αρχές, περιορισμούς και βασικές λειτουργίες
- Ανεξάρτητα της γλώσσας προγραμματισμού, του πρωτοκόλλου επικοινωνίας ή του είδους των δεδομένων

ΣΥΝΟΠΤΙΚΑ

Τέσσερις βασικές έννοιες:

- Resources, Representations, Requests, Responses

Έξι βασικές αρχές:

- Client-server, Stateless, Cacheable, Layered System, Uniform Interface, Code on demand (προαιρετικά)

Οφέλη

- Η σωστή χρήση των RESTful αρχών βελτιώνει όλα τα σημαντικά χαρακτηριστικά μιας αρχιτεκτονικής:
 - Απόδοση
 - Κλιμάκωση
 - Απλότητα
 - Επεκτασιμότητα
 - Αξιοπιστία

RESTful Web Service (API)

Αποτελείται από:

- Ένα HTTP base URL (REST endpoint)
- Ένα ή περισσότερα MIMEType για τα representations
- HTTP Methods: GET, PUT ή PATCH, POST, DELETE

Παράδειγμα

- Έστω μια συλλογή (collection) από στοιχεία (items).
- Υπάρχουν δύο βασικά endpoints:
 - `[baseUrl]/items` : Ενέργειες στη συλλογή (εν συνόλω).
 - `[baseUrl]/items/id` : Ενέργειες στο στοιχείο (ειδικά).
- REST Calls
 - `GET [baseUrl]/items`
 - `PUT [baseUrl]/items/32`

Συλλογές

| HTTP Method | Ενέργεια |
|-------------|---|
| GET | Εμφάνιση της λίστας με τα elements του collection. |
| PUT | Αντικατάσταση του υπάρχοντος collection με νέο collection (αν υποστηρίζεται). |
| POST | Προσθήκη νέου element στο υπάρχον collection. |
| DELETE | Διαγραφή του υπάρχοντος collection (αν υποστηρίζεται). |

Στοιχεία

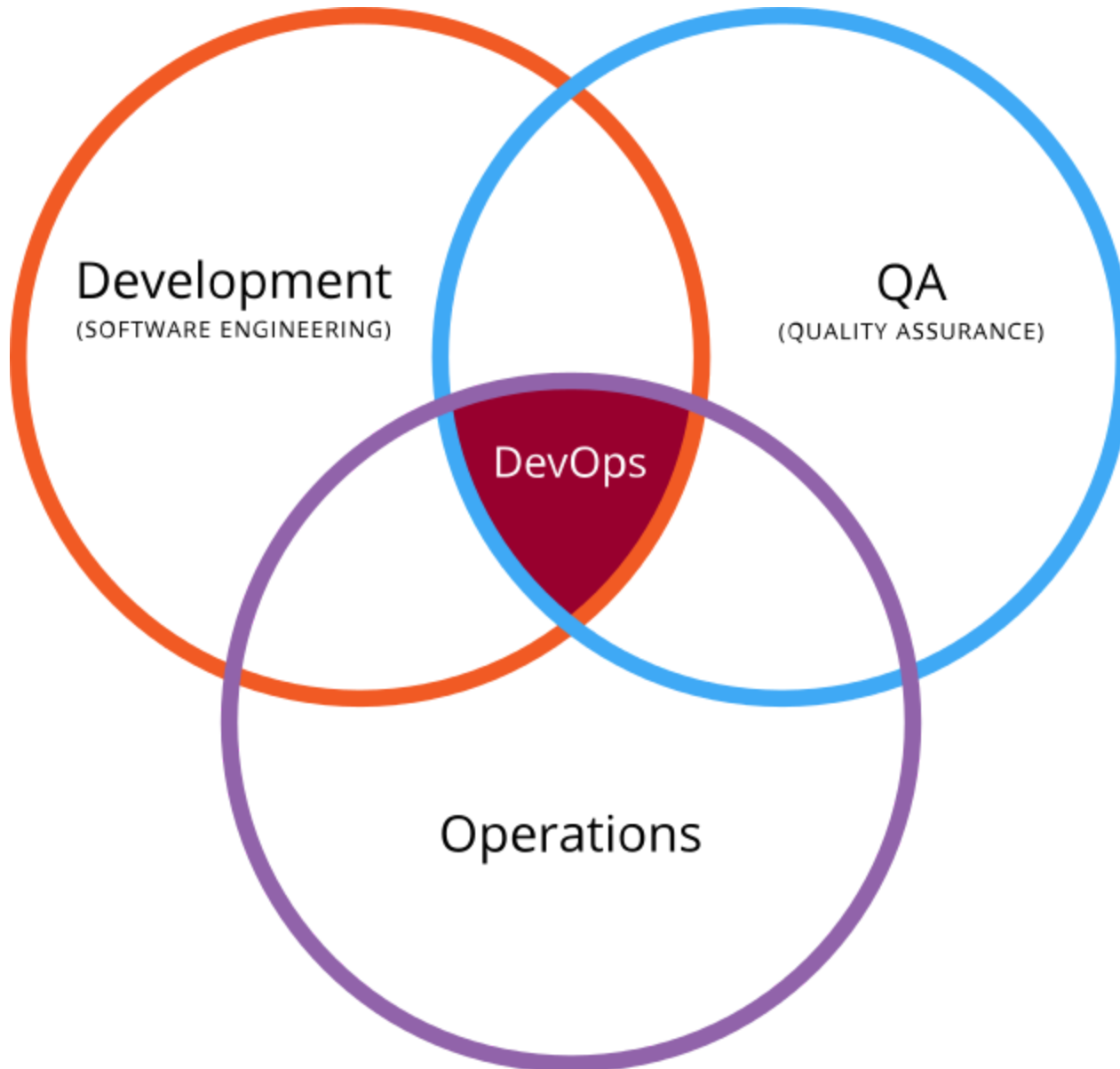
| HTTP Method | Ενέργεια |
|-------------|---|
| GET | Ανάκτηση μιας αναπαράστασης του συγκεκριμένου item. |
| PUT ή PATCH | Τροποποίηση του συγκεκριμένου item. |
| POST | Δε χρησιμοποιείται ευρέως για items. |
| DELETE | Διαγραφή του συγκεκριμένου item. |

Microservices

Κωδικοποιημένα

Microservices = SOA + Unix Principles + Agile + DevOps

DevOps



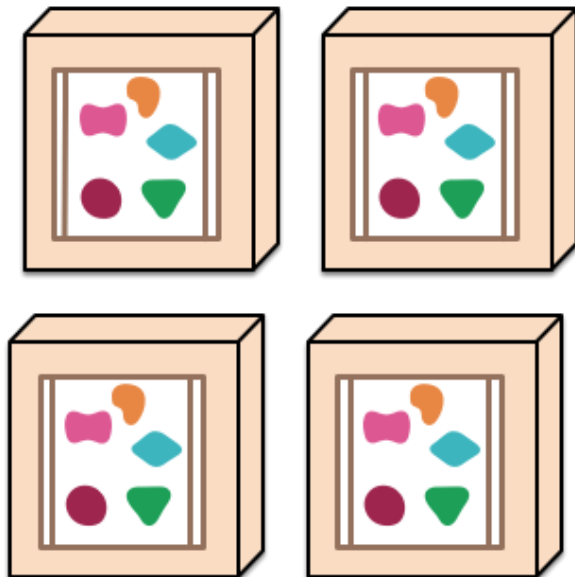
Monolithic Server-side Applications

- A single logical executable
- Cloud deployment issues:
 - Small changes -> rebuild and redeploy the whole app
 - Scalability -> scale it all or not at all

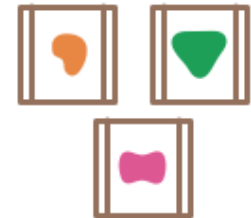
A monolithic application puts all its functionality into a single process...



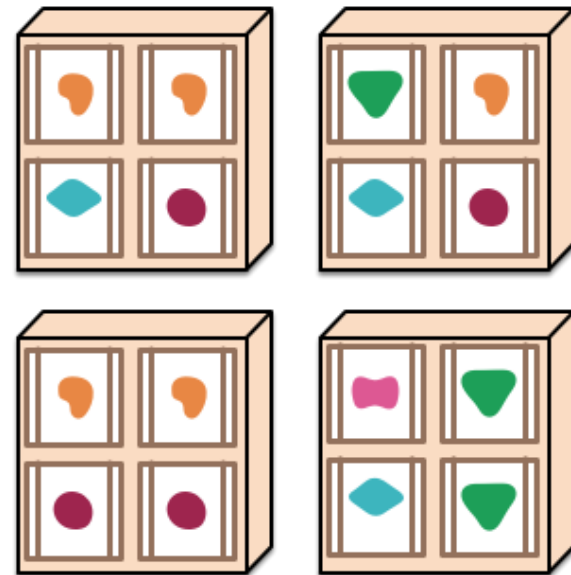
... and scales by replicating the monolith on multiple servers



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



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



Χαρακτηριστικά

- Componentization via services
 - As in SOA
- Organized around business capabilities
 - Cross-functional teams, as in Agile
- Products not projects
 - A team should own a product
 - You build it, you run it
- Smart end-points, dumb pipes
 - Simple communication (Unix-style)

- Decentralized governance
 - Each team governs the implementation details of its product
- Decentralized data management
 - Different datastore per service/app
- Infrastructure automation
 - Build & test automation
 - Continuous delivery & deployment
- Design for failure
 - Sophisticated monitoring & logging

Devops toolchain for Microservices

