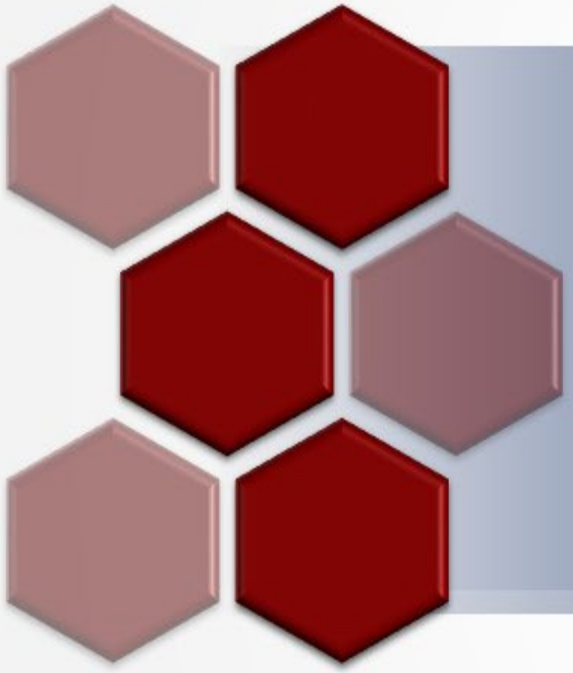
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ



Πληροφοριακά Συστήματα

Περιεχόμενα

- Σύνοψη προηγούμενης διάλεξης
- Λογική Ενοποιημένη Διεργασία
- Ανάπτυξη συστημάτων συνιστωσών λογισμικού
  - Version Control
  - Σύνθεση των ΠΣ που βασίζονται σε συνιστώσες
  - Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών
- Υπηρεσιοστρεφής ανάπτυξη συστημάτων
  - Η χρήση των υπηρεσιών ιστού
- DevOps

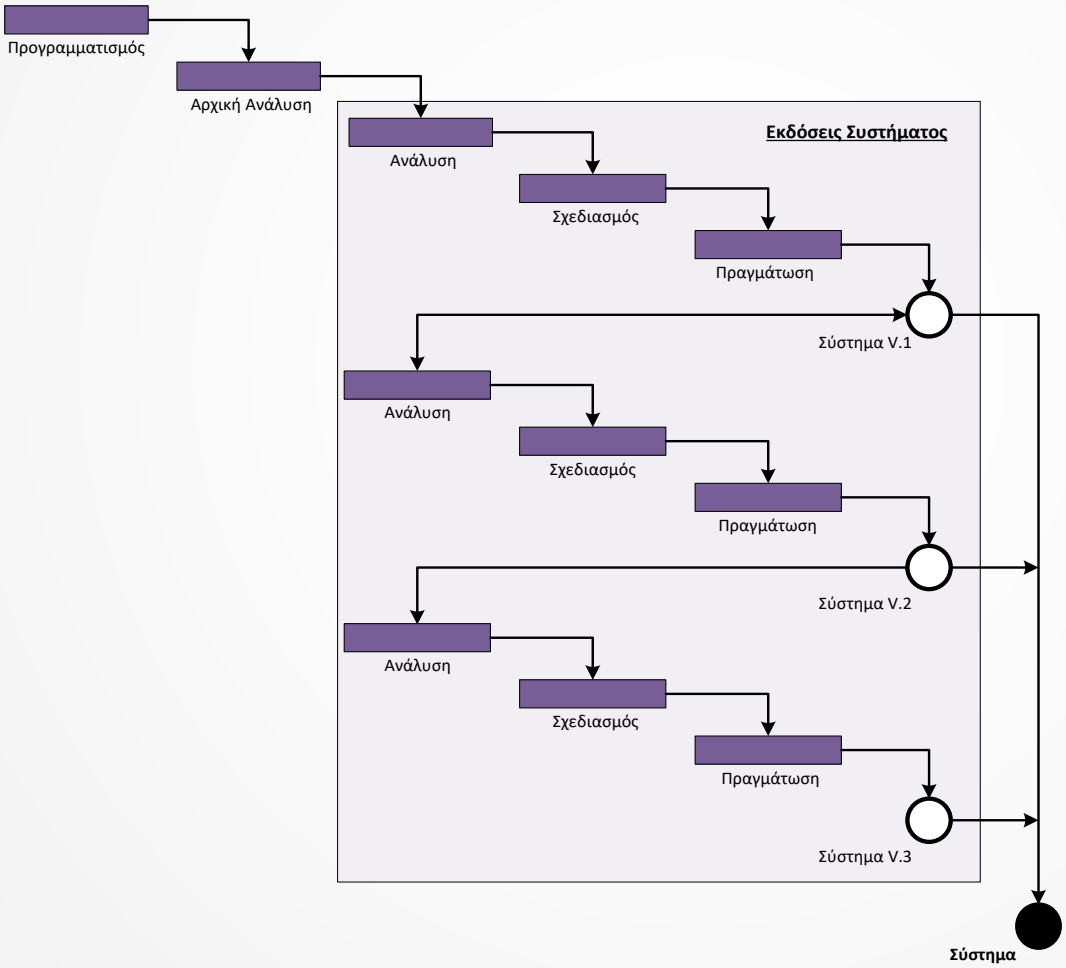
Ταχεία ανάπτυξη συστημάτων (ΤΑΣ)

- Μερικά είδη (ή παραλλαγές) της ΤΑΣ προσπαθούν να προσαρμόζονται σε πιθανές τροποποιήσεις των επιχειρησιακών διεργασιών με εκτέλεση όλων των φάσεων της ανάπτυξης του συστήματος σχεδόν ταυτόχρονα.
- Τούτο συμβαίνει σε διάφορες μεθοδολογίες που αποσκοπούν στη γρήγορη ανάπτυξη συστημάτων προς αξιολόγηση από τους χρήστες, όπως:
  - η **μεθοδολογία ταχείας ανάπτυξης πρωτοτύπων εφαρμογών** *(Prototyping RAD)* και
  - Η **μεθοδολογία ευέλικτης ανάπτυξης συστημάτων** *(Agile Systems Development Methodology)*.

Ταχεία ανάπτυξη συστημάτων (ΤΑΣ)

- Η γενική μεθοδολογία ΤΑΣ εισήγαγε τη χρήση προηγμένων εργαλείων ανάπτυξης συστημάτων, όπως:
  - οι **γεννήτριες κώδικα** *(low -> zero / no code programming)* και
    - Google AppMaker -> AppSheet, Microsoft PowerApps
  - οι **οπτικές γλώσσες προγραμματισμού**.
    - NodeRED
- Με τη χρήση αυτών των εργαλείων:
  - επιταχύνεται η διαδικασία ανάπτυξης του συστήματος και,
  - σε κάποιο βαθμό, παράγεται υψηλότερη ποιότητα κώδικα αναφορικά με τη δόμηση και την ικανοποίηση των απαιτήσεων.
- Η επιτάχυνση της παράδοσης του συστήματος, τουλάχιστον σε μία προχωρημένη πρωτότυπη μορφή, δίνει τη δυνατότητα στους χρήστες, να τροποποιούν και να επεκτείνουν τις αρχικές απαιτήσεις τους με αποτέλεσμα να βελτιώνονται τα χαρακτηριστικά του

ΤΑΣ: Μέθοδος Σταδιακής Ανάπτυξης Συστημάτων



ΤΑΣ: Ανάπτυξη συστημάτων από τελικούς χρήστες

- Στα παραπάνω μειονεκτήματα πρέπει να προστεθεί και το βασικό μειονέκτημα των ενδεχόμενων αποσπασματικών βάσεων δεδομένων που έχουν κατασκευαστεί χωρίς ενιαίο σχεδιασμό και/ή χωρίς ενιαίους κανόνες με αποτέλεσμα να δυσχεραίνεται η διαλειτουργικότητα των δεδομένων σε σημασιολογικό και σε τεχνικό επίπεδο. Το γεγονός αυτό είναι σημαντικά επιβλαβές για την διαδικασία λήψης αποφάσεων στον οργανισμό, ιδιαίτερα στη σύγχρονη εποχή κατά την οποία η ύπαρξη ομογενοποιημένων (κατ' επιθυμία) και ανοικτών δεδομένων έχει καταστεί σημαντικός παράγων στη διαδικασία λήψης εδραιωμένων αποφάσεων με βάση σύγχρονα μαθηματικά, κυρίως, μοντέλα.

Design phase: datastore(s) DECISION – ερώτημα για sample datasets

- Choices (2 baseis dedomenon + 1 datastore):
  1. Transactions (OLTP): SQL
  2. Analytics (OLAP): NoSQL
  3. Archive: Data lakes (object store: CEPH)
     - ETL: Extract (1) Transform Load (2)
- Choices (basei arithmou xriston -> arithmo nodes -> SQL / NoSQL):
  - MySQL (SQL database)
  - MongoDB (NoSQL database)
    - name: Nikos
    - tel: 21041..
    - add: [
    - home: gr. lampraki
    - office: kissamou
    -]

UserID	Name	Tel
1	Nikos	21041…
4	Maria	21098…

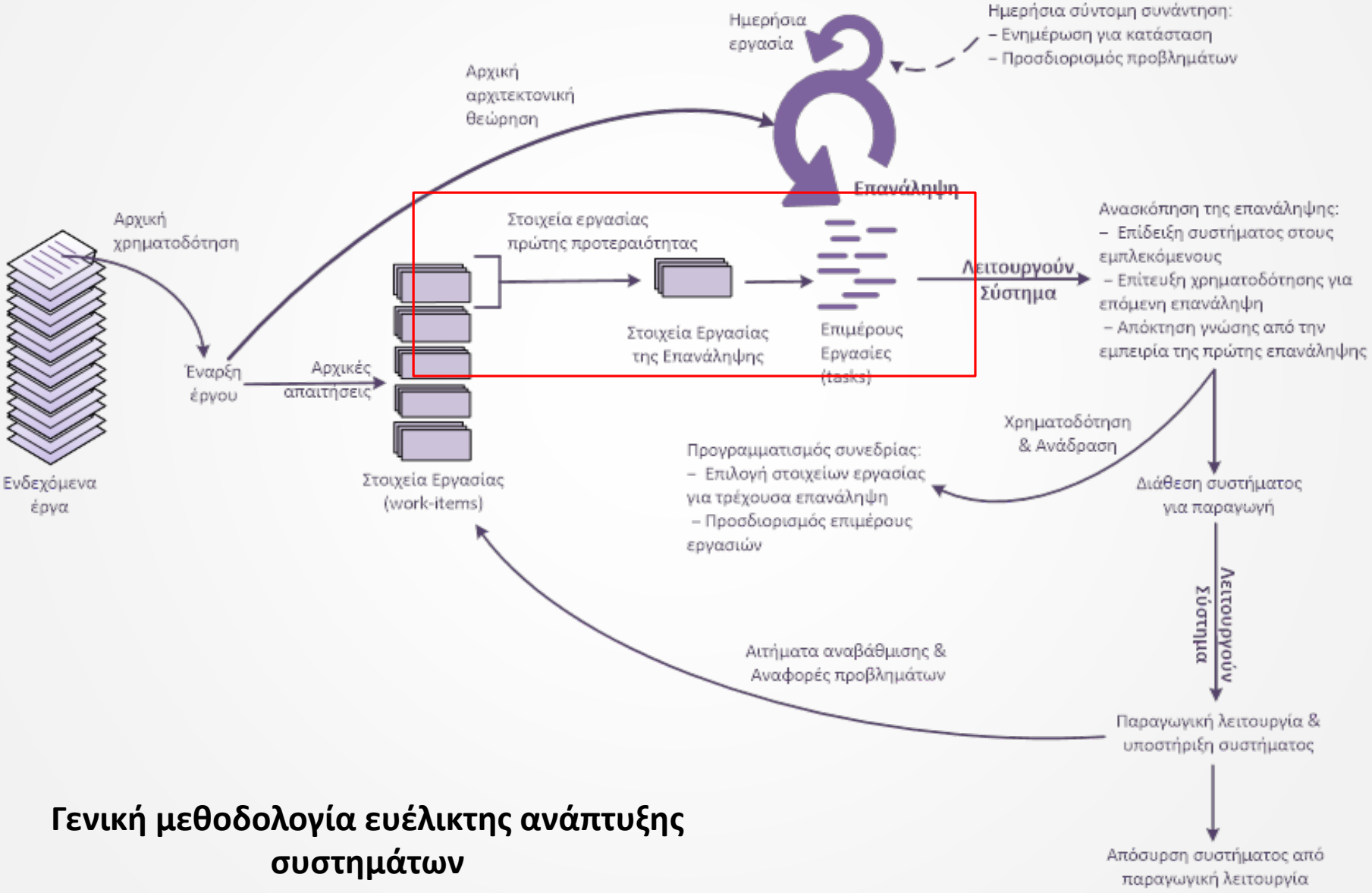
UserID	Home Address	Office Address
1	Gr. Lampraki	Kissamou …

Ευέλικτη ανάπτυξη συστημάτων

Οι περισσότερες μεθοδολογίες ευέλικτης ανάπτυξης συστημάτων **αποσκοπούν στην ελαχιστοποίηση του κινδύνου αποτυχίας αναφορικά με την ανάπτυξη και αποδοχή του λογισμικού**.

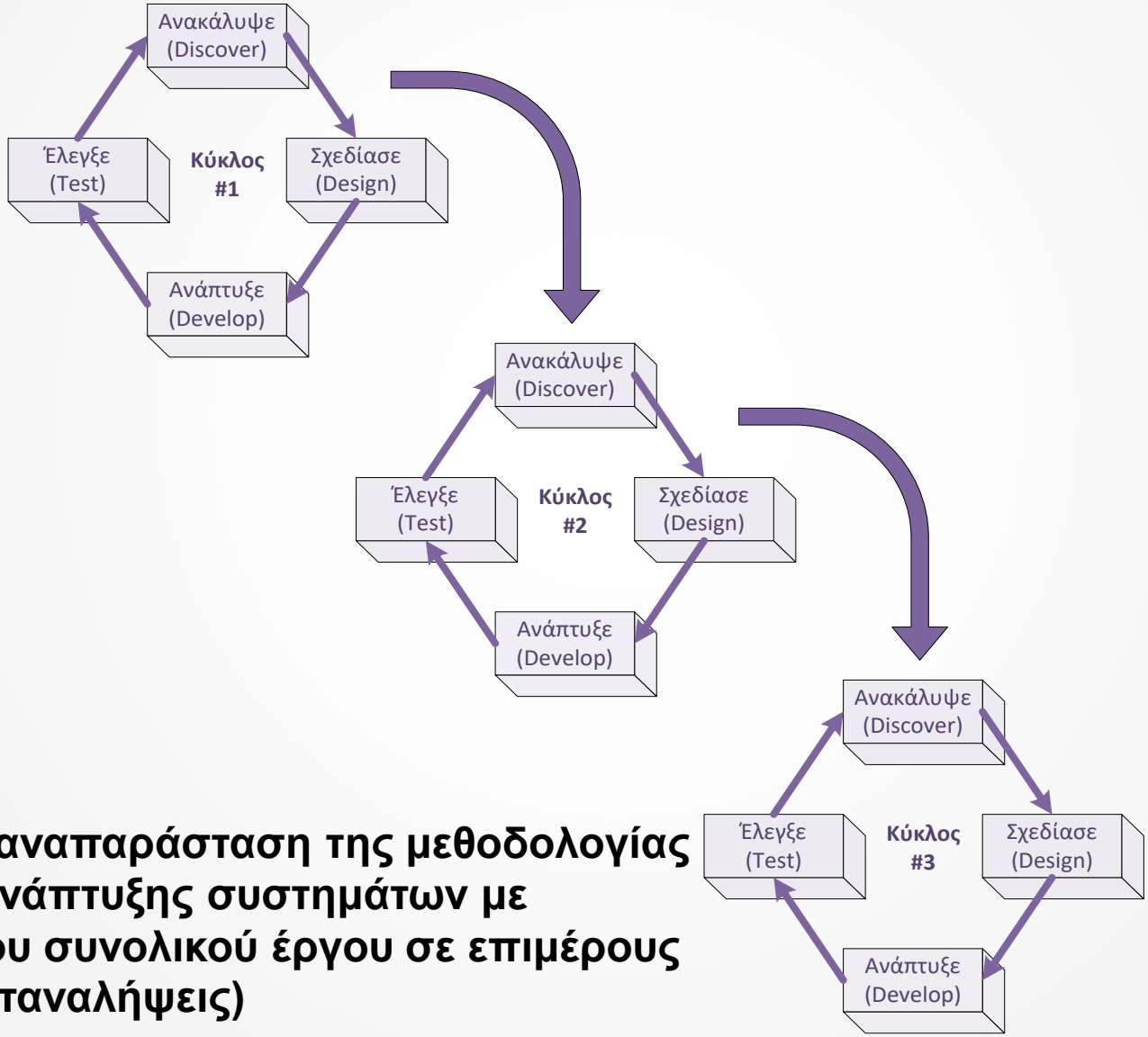
- Αναπτύσσουν το λογισμικό σε **μικρά χρονικά διαστήματα** (επαναλήψεις), διάρκειας από μία έως 4 εβδομάδες.
- Κάθε επανάληψη προσομοιάζει με ένα μικρό έργο ανάπτυξης συστήματος και περιλαμβάνει τις ακόλουθες εργασίες που είναι αναγκαίες για την κατασκευή μιας ελάχιστης επαύξησης της λειτουργίας και λειτουργικότητας του συστήματος.
  - του προγραμματισμού *(planning)*,
  - της ανάλυσης απαιτήσεων *(requirements analysis)*,
  - του σχεδιασμού *(design)*,
  - της κωδικοποίησης *(coding)*,
  - του ελέγχου ή των δοκιμών *(testing)*, και
  - της τεκμηρίωσης *(documentation)*.

Ευέλικτη ανάπτυξη συστημάτων



Γενική μεθοδολογία ευέλικτης ανάπτυξης συστημάτων

Ευέλικτη ανάπτυξη συστημάτων



Συνοπτική αναπαράσταση της μεθοδολογίας ευέλικτης ανάπτυξης συστημάτων με διάκριση του συνολικού έργου σε επιμέρους κύκλους (επαναλήψεις)

Ευέλικτη ανάπτυξη συστημάτων

- Τα σύνολα αρχών προσδιορίζονται αυθαίρετα, στο πλαίσιο των βασικών επιταγών της ευέλικτης ανάπτυξης, ανάλογα με τις εμπειρίες και τις ικανότητες (δεξιότητες) αυτού που τα προτείνει.
- Μια προσέγγιση επέκτασης των τεσσάρων αξιών της ευέλικτης ανάπτυξης σε ένα σύνολο αρχών για εφαρμογή στην ανάπτυξη συστημάτων είναι η ακόλουθη:
  - Η ενεργή εμπλοκή των χρηστών στην διαδικασία ανάπτυξης του συστήματος είναι επιβεβλημένη.
  - Η ομάδα ανάπτυξης του συστήματος πρέπει να διαθέτει την δικαιοδοσία να λαμβάνει αποφάσεις.
  - Οι απαιτήσεις εξελίσσονται και διαφοροποιούνται, πιθανώς, αλλά το χρονοδιάγραμμα παραμένει σταθερό.
    - 1. Planned (change) requests
    - 2. Add team members
  - Οι απαιτήσεις συλλέγονται σε υψηλό επίπεδο αφαίρεσης και είναι επιφανειακές και οπτικά αντιληπτές.

Ευέλικτη ανάπτυξη συστημάτων

Οι εμπνευστές των ευέλικτων μεθόδων και πρακτικών δημοσίευσαν τις κοινές τους αξίες και αρχές για την ανάπτυξη συστημάτων λογισμικού στο Μανιφέστο για την Ευέλικτη Ανάπτυξη Λογισμικού *(Agile Software Development)*. **Τα δώδεκα βασικά σημεία που περιλήφθηκαν στο μανιφέστο ευελιξίας είναι**:

i. Πρέπει να υπάρχει ικανοποίηση των πελατών μέσω της κατασκευής γρήγορων και συνεχών επαυξήσεων *(increments)*.

ii. Πρέπει να εγκατασταθεί η πρώτη επαύξηση εντός μερικών εβδομάδων και ολόκληρο το σύστημα εντός μερικών μηνών.

iii. Πρέπει οι ομάδες πελατών και κατασκευαστών να εργάζονται από κοινού καθημερινά και καθ’ όλη τη διάρκεια του έργου.

iv. Πρέπει οι ομάδες πελατών και κατασκευαστών να πραγματοποιούν συναντήσεις πρόσωπο-με-πρόσωπο.

Ευέλικτη ανάπτυξη συστημάτων

v. Πρέπει να είναι καλοδεχούμενες τυχόν απαιτήσεις που να προτείνονται από τους χρήστες ακόμη και στα τελευταία στάδια του έργου.

vi. Πρέπει να υπάρχει εμπιστοσύνη και σεβασμός μεταξύ των μελών της ομάδας κατασκευαστών.

vii. Πρέπει να μετράται η ταχύτητα εκτέλεσης του έργου μετά την παράδοση κάθε επαύξησης.

viii. Πρέπει να δίδεται έμφαση στον καλό σχεδιασμό για να αυξηθεί η ευελιξία.

ix. Πρέπει να υπάρχει αυτό-οργάνωση για την καλύτερη αρχιτεκτονική και τον καλύτερο σχεδιασμό του συστήματος.

x. Πρέπει να υπάρχει προσαρμογή και συντονισμός σύμφωνα με την κατάσταση και τις συνθήκες.

xi. Πρέπει να ακολουθείται η αρχή τη απλότητας *(keep it simple - KIS)* καθ' όλη τη διεργασία ανάπτυξης του συστήματος.

xii. Πρέπει να υπάρχει συνεπής εργασία από την αρχή μέχρι το τέλος των έργων ευέλικτης ανάπτυξης.

- Η φιλοσοφική αυτή θεώρηση σημαίνει ότι:
  - **κάθε ομάδα ανάπτυξης λογισμικού διαθέτει διαφορετικά ταλέντα και διαφορετικές δεξιότητες, οπότε πρέπει να χρησιμοποιεί διαδικασίες που είναι μοναδικά προσαρμοσμένες σ' αυτήν**.
  - πρέπει να **ελαχιστοποιηθεί η πολυπλοκότητα των διαδικασιών αφού οι διαδικασίες δεν είναι ιδιαίτερα σημαντικές** για την επιτυχία του έργου.

- Η μεθοδολογία δυναμικής ανάπτυξης συστημάτων *(Dynamic Systems Development - DSD)* αποτελεί μια εξέλιξη των πρακτικών της ταχείας ανάπτυξης συστημάτων *(Rapid Application Development - RAD)*.
- Η μεθοδολογία DSD:
  - Βασίζεται στη φιλοσοφική θεώρηση ότι **“τίποτα δεν μπορεί να κατασκευάζεται τέλεια από την πρώτη φορά”** και
  - Θεωρεί την ανάπτυξη συστημάτων ως μια επαναληπτική και διερευνητική προσπάθεια.
  - Θεωρείται ως η μεθοδολογία που προβλέπει εκπαίδευση και τεκμηρίωση περισσότερο από κάθε άλλη μεθοδολογία ευέλικτης ανάπτυξης συστημάτων.

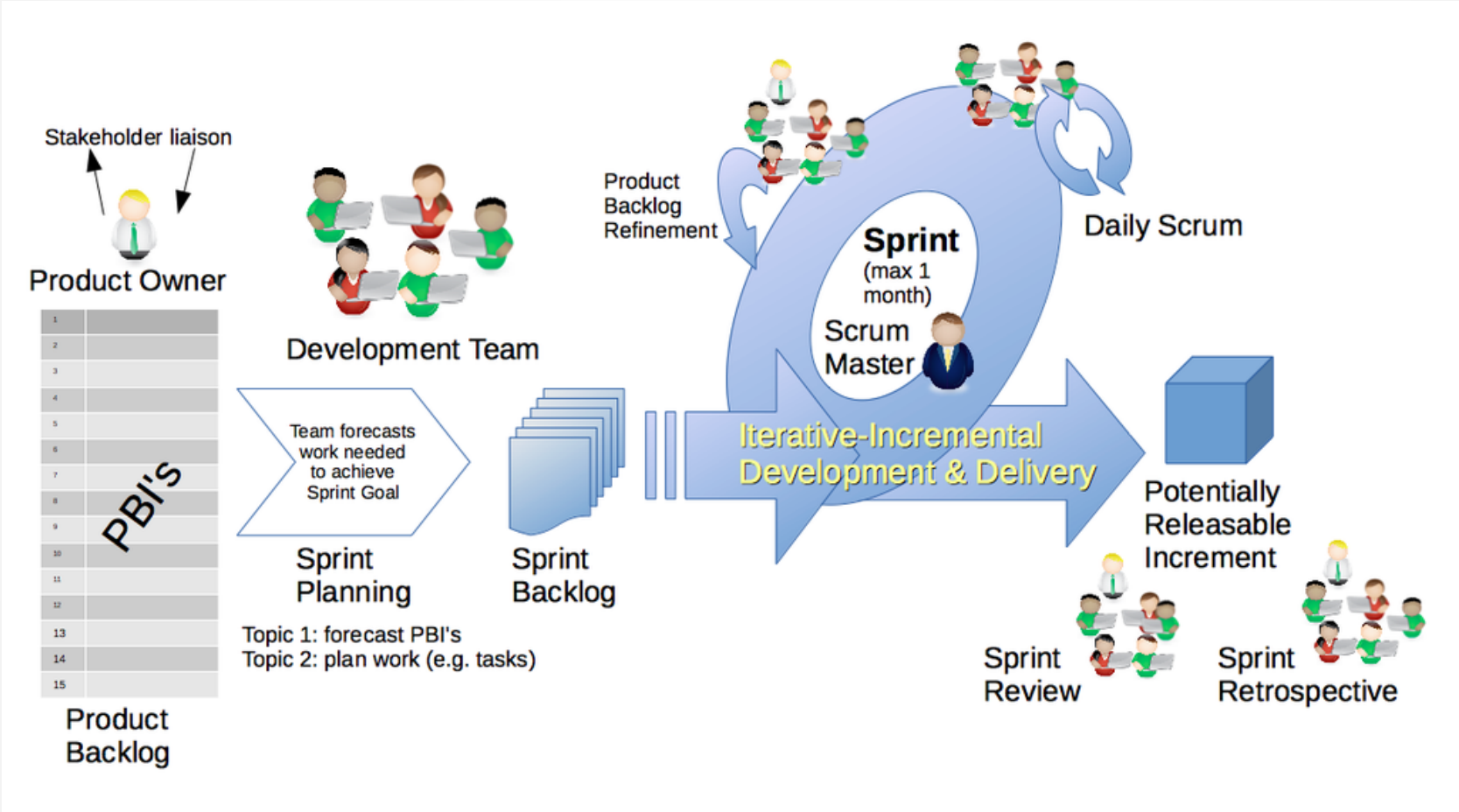
- Η μεθοδολογία DSD είναι πιθανώς η πρωταρχική μεθοδολογία ευέλικτης ανάπτυξης συστημάτων και, ενώ υπήρχε πριν την επινόηση του όρου *"agile* - ευέλικτη", βασίζεται αποκλειστικά στις λεγόμενες ευέλικτες αρχές που ανάγονται στις ακόλουθες εννέα:
  - Ενεργή συμμετοχή των χρηστών
  - Παροχή στις ομάδες ανάπτυξης του συστήματος της δικαιοδοσίας για τη λήψη αποφάσεων
  - Εστίαση στη συχνή παράδοση νέων εκδόσεων του υπό ανάπτυξη συστήματος (των προϊόντων λογισμικού)
  - Χρησιμότητα του συστήματος ως βασικό κριτήριο για την αποδοχή του

- Οι ομάδες Scrum αποτελούνται συνήθως από 7 +/- 2 μέλη και δεν έχουν επικεφαλής ομάδας να αναθέτουν καθήκοντα ή να αποφασίζουν πώς λύνεται ένα πρόβλημα.
- Η ομάδα ως μονάδα αποφασίζει πώς να αντιμετωπίσει θέματα και να λύσει προβλήματα.
- Υπάρχουν τρεις βασικοί ρόλοι σε μια ομάδα Scrum:
  - **Ο (product owner) ιδιοκτήτης προϊόντος** είναι ο βασικός παράγοντας του έργου - συνήθως ένας εσωτερικός ή εξωτερικός πελάτης ή ένας εκπρόσωπος του πελάτη. Υπάρχει μόνο ένας ιδιοκτήτης προϊόντων που μεταφέρει τη συνολική αποστολή και το όραμα του προϊόντος που κατασκευάζει η ομάδα.
  - Ο **Scrum Master** είναι ο ηγέτης των υπηρεσιών του ιδιοκτήτη προϊόντων, της ομάδας ανάπτυξης και της οργάνωσης. Χωρίς ιεραρχική εξουσία πάνω στην ομάδα, αλλά περισσότερο ως διευκολυντής, ο Scrum Master εξασφαλίζει ότι η ομάδα τηρεί τη θεωρία, τις πρακτικές και τους κανόνες του Scrum. Αυτό μπορεί να περιλαμβάνει την αφαίρεση των εμποδίων, τη διευκόλυνση των συναντήσεων και τη βοήθεια προς τον ιδιοκτήτη του προϊόντος, για την περιποίηση.
  - Η **Ομάδα Ανάπτυξης** είναι μια αυτο-οργανωτική, διαλειτουργική ομάδα εξοπλισμένη με όλες τις δεξιότητες για να παραδώσει φορτωτικές προσαυξήσεις κατά την ολοκλήρωση κάθε Sprint.

- Backlog: Είναι το πιο σημαντικό έγγραφο της διαδικασίας, όπου περιγράφεται κάθε απαίτηση για ένα σύστημα, ένα έργο ή ένα προϊόν. Το backlog μπορεί να θεωρηθεί ως λίστα υποχρεώσεων που αποτελείται από στοιχεία εργασίας, καθένα από τα οποία παράγει ένα παραδοτέο με επιχειρηματική αξία. Τα στοιχεία του backlog ταξινομούνται με βάση την σημαντικότητά τους από τον ιδιοκτήτη του προϊόντος.
- Sprint Backlog: Είναι το τμήμα με τα στοιχεία του backlog που αναμένεται να ολοκληρωθούν σε ένα συγκεκριμένο sprint.
- Increment: Είναι το σύνολο όλων των στοιχείων του backlog που έχουν ολοκληρωθεί από στην τελευταία έκδοση του λογισμικού.

- Sprint: Ένα sprint είναι μια χρονική περίοδος κατά την οποία η συγκεκριμένη εργασία ολοκληρώνεται και είναι έτοιμη για επανεξέταση. Η συνήθης διάρκεια είναι 2-4 εβδομάδες.
- Sprint Planning: Οι συναντήσεις ομάδων προγραμματισμού που καθορίζουν ποιες λειτουργίες θα παραδοθούν στο συγκεκριμένο sprint και πώς θα επιτευχθεί η εργασία.
- Daily Scrum ή Stand-up: Είναι μια σύντομη συνάντηση επικοινωνίας (όχι περισσότερο από 15 λεπτά) στην οποία κάθε μέλος της ομάδας καλύπτει γρήγορα και διαφανώς την πρόοδο από την τελευταία στάση, την προγραμματισμένη εργασία πριν από την επόμενη συνάντηση και τυχόν εμπόδια που ενδέχεται να εμποδίζουν την πρόοδό της.
- Sprint Review: Είναι η εκδήλωση επίδειξης για να παρουσιάσει το έργο που ολοκληρώθηκε κατά τη διάρκεια του sprint. Ο ιδιοκτήτης προϊόντος ελέγχει το έργο σύμφωνα με προκαθορισμένα κριτήρια αποδοχής και είτε δέχεται είτε απορρίπτει το έργο.
- Sprint Retrospective: είναι η τελική συνάντηση της ομάδας στο sprint για να καθορίσει τι πήγε καλά, τι δεν πήγε καλά και πώς μπορεί να βελτιωθεί η ομάδα στο επόμενο.

Ευέλικτη ανάπτυξη συστημάτων: Η μεθοδολογία Scrum



- Η μεθοδολογία ακραίου προγραμματισμού (μεθοδολογία XP):
  - Αποσκοπεί στην παροχή της δυνατότητας κατασκευής λογισμικού ακόμη και εντός ενός ασταθούς περιβάλλοντος επιτρέποντας σημαντική ευελιξία στη διεργασία μοντελοποίησης του συστήματος.
  - Ανήκει στην κατηγορία των ευέλικτων μεθοδολογιών ανάπτυξης συστημάτων και μπορεί να θεωρηθεί είτε ως μια νέα προσέγγιση είτε ως μια επέκταση της μεθόδου των εξελικτικών πρωτοτύπων ή των μεθοδολογιών ταχείας ανάπτυξης εφαρμογών.
- Αντί της τυπικής παράδοσης των απαιτήσεων από τους χρήστες στους κατασκευαστές, **οι χρήστες ενθαρρύνονται να παρέχουν παραδείγματα αναφορικά με τη χρησιμοποίηση του συστήματος** σε συγκεκριμένες επιχειρησιακές καταστάσεις.
- Κύριος στόχος της μεθοδολογίας XP είναι η μείωση του κόστους από αλλαγές των απαιτήσεων των χρηστών.

Περιεχόμενα

- Σύνοψη προηγούμενης διάλεξης
- Λογική Ενοποιημένη Διεργασία
- Ανάπτυξη συστημάτων συνιστωσών λογισμικού
  - Version Control
  - Σύνθεση των ΠΣ που βασίζονται σε συνιστώσες
  - Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών
- Υπηρεσιοστρεφής ανάπτυξη συστημάτων
  - Η χρήση των υπηρεσιών ιστού
- DevOps

Λογική Ενοποιημένη Διεργασία

- Η Λογική Ενοποιημένη Διεργασία – ΛΕΔ *(Rational Unified Process - RUP)* ανήκει στην κατηγορία των εξελικτικών μεθοδολογιών και θέτει ως κύριο **στόχο-μέλημα** τη **μείωση των κινδύνων αποτυχίας του έργου της ανάπτυξης ΠΣ**.
- Επιτρέπει να λαμβάνονται υπόψη τυχόν αλλαγές στις απαιτήσεις κατά τη διάρκεια ανάπτυξης του συστήματος.
- Ενθαρρύνεται η σταδιακή ολοκλήρωση των ενοτήτων του συστήματος (αντικειμένων, συνιστωσών ή εφαρμογών) σε αντίθεση με την ολοκλήρωση στα τελικά στάδια του έργου υπό μορφή *“big bang”*.
- Μετριάζονται οι κίνδυνοι αποτυχίας του έργου λόγω του ότι εντοπίζονται νωρίς, και όχι κατά τη κατά τη διάρκεια της ολοκλήρωσης, οπότε αντιμετωπίζονται επιτυχώς.

Λογική Ενοποιημένη Διεργασία

- Η μεθοδολογία RUP αποτελεί ένα από τα βασικά εργαλεία ανάπτυξης συστημάτων της εταιρείας IBM (αρχικά της, εξαγορασθείσης από την IBM, εταιρείας *RATIONAL*) και συνιστά μια από τις πιο διαδεδομένες και ευρέως χρησιμοποιούμενες μεθοδολογίες παγκοσμίως.
- Η μεθοδολογία RUP διαθέτει τα ακόλουθα βασικά χαρακτηριστικά:
  - Ακολουθεί το μοντέλο των εξελικτικών (επαυξητικών) μεθοδολογιών ανάπτυξης συστημάτων βασιζόμενη στη σταδιακή και επαναληπτική ανάπτυξη του συστήματος.
  - **Χρησιμοποιεί τη UML**, η οποία παρέχει μια πλούσια συλλογή διαγραμμάτων για την μοντελοποίηση διαφόρων θεωρήσεων του συστήματος (λειτουργικών, στατικών και δυναμικών), γεγονός το οποίο της προσδίδει το πρόσθετο πλεονέκτημα της παραγωγής τεκμηριωτικού υλικού σε ενιαία γλώσσα.

Λογική Ενοποιημένη Διεργασία

- Παρέχεται η δυνατότητα στη διοίκηση του έργου να πραγματοποιεί τακτικές αλλαγές στο τελικό προϊόν λόγω της επαναληπτικής ανάπτυξης. Έτσι, επιτρέπεται η διάθεση του λογισμικού με μειωμένη λειτουργικότητα σε σύντομο χρόνο προκειμένου είτε να αντιπαραταχθεί ο οργανισμός σε αντίστοιχες κινήσεις του ανταγωνισμού είτε να επιλεγεί άλλος κατασκευαστής για μια συγκεκριμένη τεχνολογία.
- Διευκολύνεται η επαναχρησιμοποίηση συνιστωσών λογισμικού λόγω της επαναληπτικής ανάπτυξης. Και τούτο διότι είναι ευκολότερο να εντοπιστούν τα κοινά μέρη όταν έχουν σχεδιαστεί και κατασκευαστεί μερικώς κατά την αρχική φάση του προγραμματισμού *(planning)*.

Λογική Ενοποιημένη Διεργασία: Φάσεις

- Η RUP αποτελείται από τέσσερις κύριες φάσεις οι οποίες εκτελούνται διαδοχικά:
  - Σύλληψη *(Inception)*,
  - Εντρύφηση *(Elaboration)*,
  - Κατασκευή *(Construction)* και
  - Μετάβαση *(Transition)*.
- Κάθε φάση έχει συγκεκριμένο σκοπό και περατώνεται σε ένα ορόσημο που σηματοδοτεί το γεγονός *(event)* και το χρονικό σημείο *(time point)* στο οποίο επιτυγχάνεται μια συγκεκριμένη πρόοδος του έργου.
- Γενικά, η μεθοδολογία RUP μπορεί να θεωρηθεί ως **ενδιάμεση λύση μεταξύ της καταρρακτοειδούς μεθοδολογίας και των μεθοδολογιών ευέλικτης ανάπτυξης συστημάτων**.

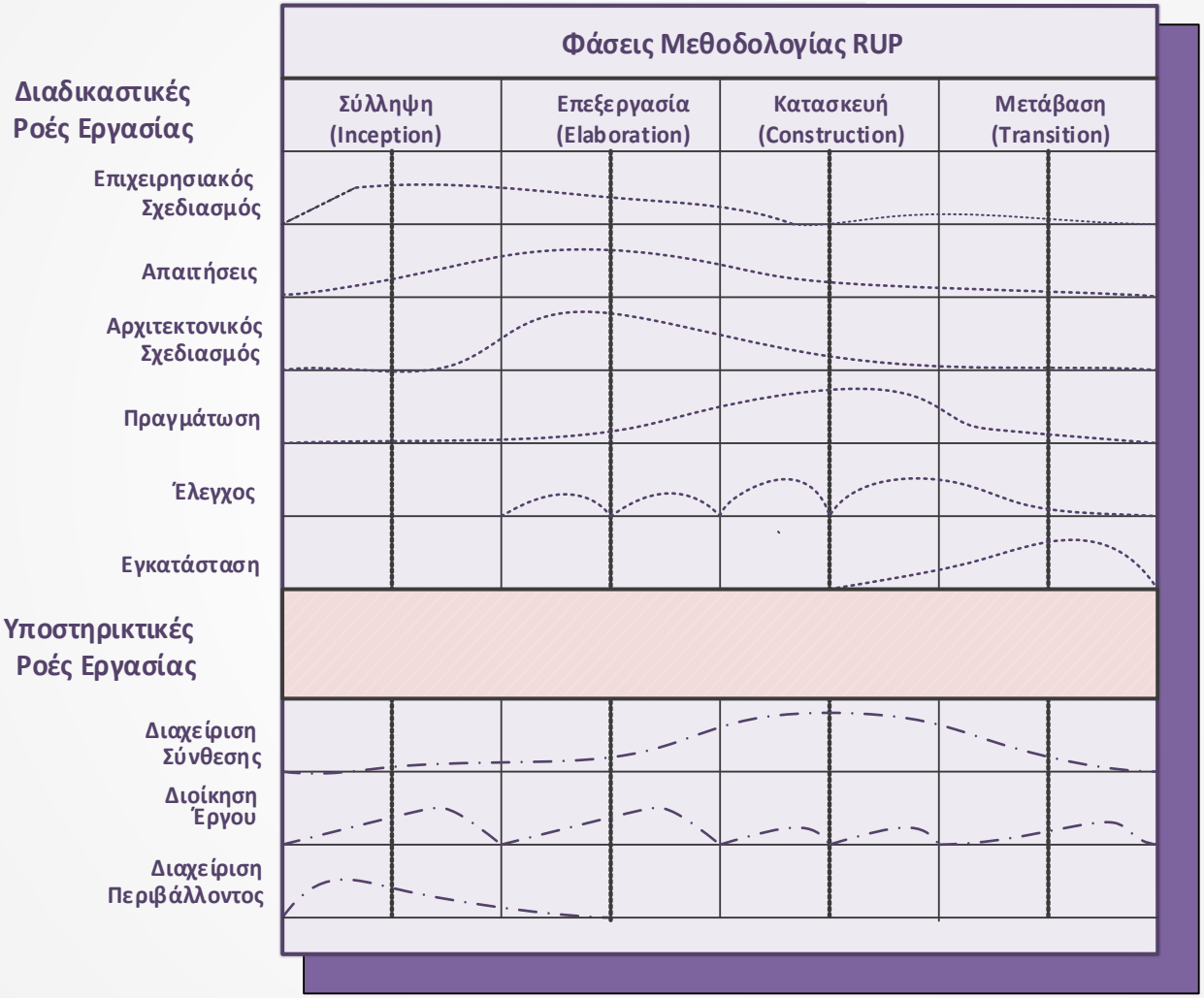
Λογική Ενοποιημένη Διεργασία: Φάσεις

- **Σύλληψη:** Ορίζεται το εύρος και το περιεχόμενο του έργου (**οριοθετείται το σύστημα**). Παρουσιάζεται η αρχική ιδέα για το υπό ανάπτυξη σύστημα στο απαιτούμενο επίπεδο λεπτομέρειας ώστε να πεισθούν οι αρμόδιοι για την προώθηση του έργου στην επόμενη φάση που είναι η φάση της εντρύφησης.
- **Εντρύφηση:** Προσδιορίζεται η μέθοδος για την πραγμάτωση του έργου, τη μοντελοποίηση των θεωρήσεων (όψεων) του συστήματος και τον ορισμό της αρχιτεκτονικής του συστήματος.
- **Κατασκευή:** Πραγματώνεται και ελέγχεται το υπό κατασκευή σύστημα σε εργαστηριακό, μη παραγωγικό, περιβάλλον. Ο έλεγχος του συστήματος ως μονάδες και στο σύνολο πραγματοποιείται ταυτόχρονα με την πραγμάτωση του συστήματος και δεν έπεται αυτής όπως συχνά συμβαίνει σε περιπτώσεις υιοθέτησης μιας παραδοσιακής προσέγγισης κύκλου ζωής.

Λογική Ενοποιημένη Διεργασία: Φάσεις

- **Μετάβαση:** Εκτελούνται όλες οι προπαρασκευαστικές εργασίες για την, κατά το δυνατόν, ομαλή και απρόσκοπτη εγκατάσταση και θέση σε παραγωγική λειτουργία του συστήματος. Εγκαθίσταται και ελέγχεται (δοκιμάζεται) το υπό κατασκευή σύστημα (σε επίπεδο μέρους και όλου) στο περιβάλλον παραγωγικής λειτουργίας και χρήσης του. Η φάση αυτή σηματοδοτεί και την έναρξη της φάσης της συντήρησης του συστήματος.

Λογική Ενοποιημένη Διεργασία



Οργάνωση για την εφαρμογή της μεθοδολογίας RUP. Εκτιμώμενος φόρτος εργασίας ανά δραστηριότητα ροών εργασίας και φάση

Λογική Ενοποιημένη Διεργασία: Φάσεις

Οι διαδικαστικές ροές εργασίας είναι:

- Η μοντελοποίηση του επιχειρησιακού περιβάλλοντος *(Business Modeling)*,
- Ο προσδιορισμός των απαιτήσεων *(Requirements Specification)*,
- Ο αρχιτεκτονικός σχεδιασμός του συστήματος *(System Architecture and Design)*,
- Η πραγμάτωση του συστήματος *(System Implementation)*,
- Ο έλεγχος του συστήματος αναφορικά με την ικανοποίηση των απαιτήσεων *(System Testing),* και
- Η εγκατάσταση στο συστήματος στο παραγωγικό περιβάλλον *(System Deployment)*.

Λογική Ενοποιημένη Διεργασία: Φάσεις

Οι υποστηρικτικές ροές εργασίας είναι:

- Η διαχείριση της σύνθεσης του συστήματος *(Configuration Management)*,
- Η διοίκηση του έργου *(Project Management),* και
- Η διαχείριση του περιβάλλοντος ανάπτυξης του συστήματος *(Development Environment Management)*.

Λογική Ενοποιημένη Διεργασία: Βέλτιστες πρακτικές

Η RUP ενσωματώνει συγκεκριμένες "βέλτιστες πρακτικές" *(best practices),* που είναι εμπειρικά δοκιμασμένες προσεγγίσεις ανάπτυξης συστημάτων και είναι οι ακόλουθες:

- Επαναληπτική διεργασία ανάπτυξης συστήματος *(Develop iteratively)*,
- Διαχείριση απαιτήσεων *(Manage requirements),*
- Χρήση αρχιτεκτονικών που βασίζονται σε συνιστώσες λογισμικού *(Use component architectures)*,
- Μοντελοποίηση λογισμικού με τη χρήση εργαλείων οπτικοποίησης *(Model visually),*
- Επαλήθευση της ποιότητας λογισμικού *(Verify software quality)*, και
- Έλεγχος των αλλαγών του συστήματος *(Control changes)*.

Λογική Ενοποιημένη Διεργασία: Βέλτιστες πρακτικές

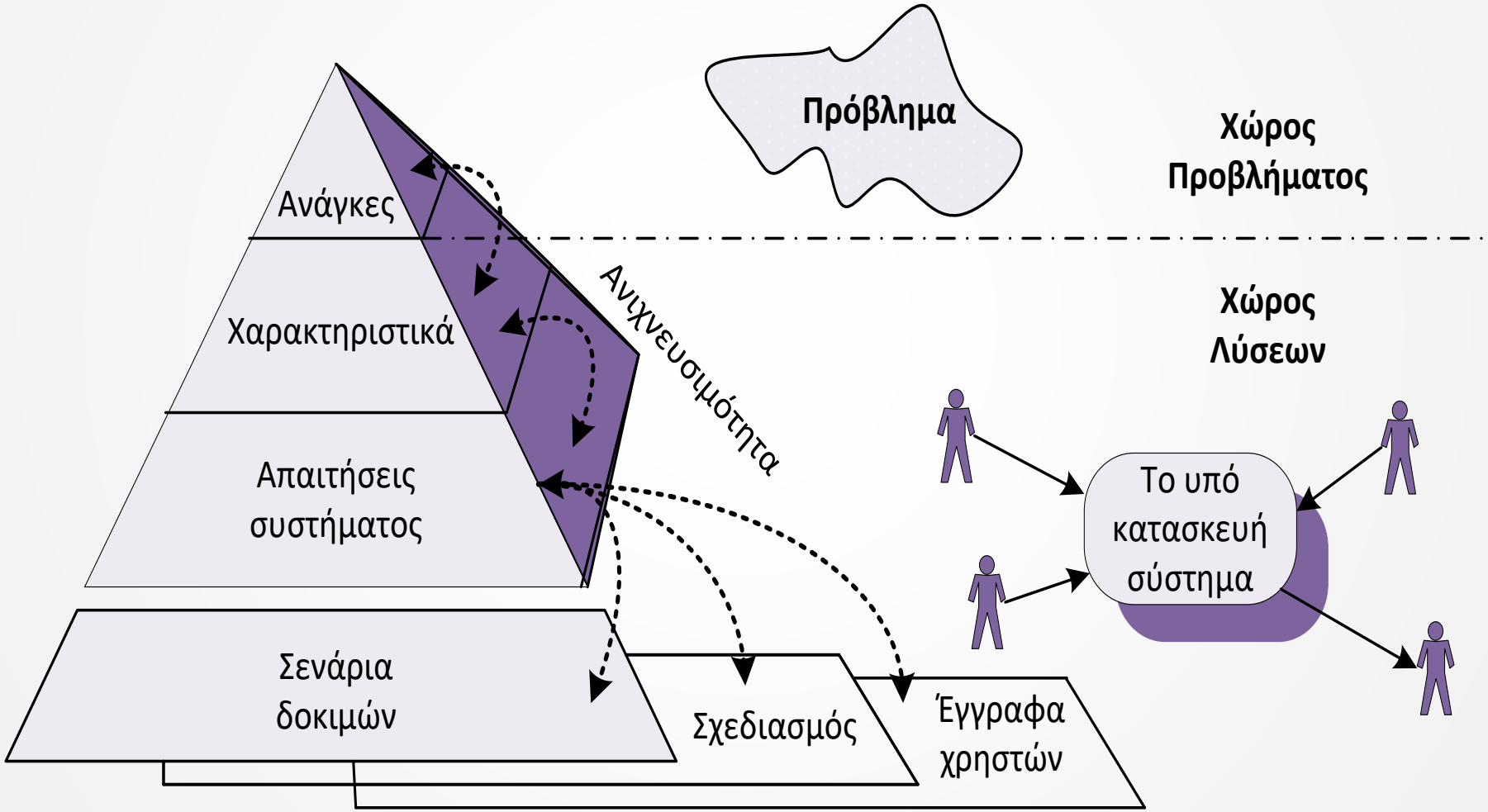
i. **Επαναληπτική διεργασία ανάπτυξης συστήματος:** Συνιστά μια τεχνική που χρησιμοποιείται προκειμένου να παράσχει την απαιτούμενη λειτουργία και λειτουργικότητα του συστήματος μέσω μιας διαδικασίας κατασκευής διαδοχικών, βελτιωμένων και επαυξημένων εκδοχών του *(system versions)*. Κάθε εκδοχή του συστήματος αναπτύσσεται εντός ενός συγκεκριμένου χρονικού διαστήματος που ονομάζεται *“επανάληψη” (iteration)* και αφορά στον ορισμό, στην ανάλυση, στο σχεδιασμό, στην κατασκευή και στη δοκιμή του συστήματος με βάση ένα σύνολο απαιτήσεων (που μπορεί να έχουν προσδιοριστεί εξαρχής ή να προέκυψαν στη συνέχεια κατά τη διεργασία ανάπτυξης στου συστήματος).

ii. **Διαχείριση απαιτήσεων:** Αφορά στη διαδικασία διασφάλισης ότι το υπό ανάπτυξη σύστημα θα καλύπτει τις ανάγκες των χρηστών (συμπεριλαμβανομένης της διοίκησης) για τους οποίους προορίζεται με βάση μιας συγκροτημένης διαδικασίας συγκερασμού διαφορετικών απόψεων στο πλαίσιο της επερχόμενης αλλαγής (π.χ. στο εργασιακό περιβάλλον και στη διαδικασία λήψης αποφάσεων) που θα επέλθει με την ανάπτυξη και λειτουργίας του ΠΣ. Συνεπώς, ο κύριος στόχος αυτής της βέλτιστης πρακτικής είναι η ελαχιστοποίηση του κινδύνου ανάπτυξης ενός ανεπιτυχούς συστήματος, δηλαδή ενός συστήματος που θα είναι δύσχρηστο και μη χρήσιμο κατά την αντίληψη των χρηστών (και συνεπώς μη αποδεκτό).

Λογική Ενοποιημένη Διεργασία: Βέλτιστες πρακτικές

- Σ’ αυτό το πλαίσιο, η RUP παρέχει μια συστηματική προσέγγιση για την **εκμαίευση**, την **οργάνωση**, την **καταγραφή**, την **αξιολόγηση**, τον **προσδιορισμό** και τη **διαχείριση** των υποκείμενων σε διαρκείς αλλαγές **απαιτήσεων ενός συστήματος**.
- Ειδικότερα, η RUP **επιβάλει τη διαχείριση των απαιτήσεων καθ’ όλη τη διάρκεια του κύκλου ζωής** που έχει επιλεγεί επιτρέποντας την τροποποίηση και/ή την επαύξησή τους σε κάθε επανάληψη.
- Με αυτόν τον τρόπο διασφαλίζεται η τελική αποδοχή του συστήματος, ως το μείζον, αν και μπορεί να επέλθει σημαντική υπέρβαση του χρονοδιαγράμματος και του προϋπολογισμού του έργου.

Λογική Ενοποιημένη Διεργασία: Βέλτιστες πρακτικές



Λογική Ενοποιημένη Διεργασία: Βέλτιστες πρακτικές

iii. **Χρήση αρχιτεκτονικών που βασίζονται σε συνιστώσες λογισμικού –** Συνιστώσα λογισμικού είναι ένα υποσύστημα του συστήματος που αντικατοπτρίζει μια συγκεκριμένη λειτουργία (απλή ή σύνθετη) και έχει αυτόνομη λειτουργικότητα.

- Η RUP ενθαρρύνει τη χρήση υπαρχόντων ή τη δημιουργία νέων συνιστωσών για τη σύνθεση ενός ολοκληρωμένου και σπονδυλωτού συστήματος.
- Η αντίστοιχη αρχιτεκτονική του συστήματος ονομάζεται αρχιτεκτονική βασισμένη σε συνιστώσες *(component-based)* και συνιστά τον πρόδρομο των υπηρεσιοστρεφών αρχιτεκτονικών *(service-oriented)* οι οποίες ενθαρρύνουν και την ενσωμάτωση των υπαρχόντων εφαρμογών ως συνιστωσών.
- Το κυριότερο χαρακτηριστικό μιας αρχιτεκτονικής που βασίζεται σε συνιστώσες είναι η ευελιξία που διαθέτει αναφορικά με την ενσωμάτωση στο σύστημα νέων ή τροποποιημένων απαιτήσεων.

iv. **Μοντελοποίηση συστήματος με τη χρήση εργαλείων οπτικοποίησης** – Εξ' ορισμού κάθε μοντέλο συνιστά μια απλοποιημένη αναπαράσταση της πραγματικότητας και στοχεύει στην παροχή μιας περιγραφής του υπό ανάπτυξη συστήματος ιδωμένη από μια συγκεκριμένη οπτική. Βασικός σκοπός της δημιουργίας μοντέλων για τα λειτουργικά, στατικά (δομικά) και δυναμικά (συμπεριφορικά) χαρακτηριστικά του υπό ανάπτυξη συστήματος είναι η υποβοήθηση της διεργασίας καλύτερης κατανόησής του. Ιδιαίτερα δε όταν το σύστημα είναι σύνθετο, όπως συχνά συμβαίνει στην πραγματικότητα, και δεν είναι εύκολη η πλήρης και σε βάθος κατανόησή του χωρίς την λειτουργική και/η διεργασιακή τμηματοποίησή του.

Λογική Ενοποιημένη Διεργασία: Βέλτιστες πρακτικές

v. **Επαλήθευση της ποιότητας λογισμικού:** Η επαλήθευση της ποιότητας του λογισμικού δεν αποτελεί ξεχωριστή διεργασία αλλά είναι ενσωματωμένη σε όλες τις δραστηριότητες της διεργασίας ανάπτυξης του συστήματος. Αναφέρεται δε σε όλους τους συμμετέχοντες σ' αυτήν (κατασκευαστές και χρήστες) χρησιμοποιώντας αντικειμενικά κριτήρια και μετρικές.

vi. **Έλεγχος των αλλαγών του συστήματος:** Σε ένα περιβάλλον ανάπτυξης συστημάτων, όπου οι αλλαγές των απαιτήσεων είναι αναπόφευκτες, πρέπει να ελέγχεται και να παρακολουθείται το *πώς* και το *πότε* ενσωματώνονται οι αλλαγές αυτές στα παραδοτέα του έργου και ποιος αναλαμβάνει την ευθύνη για την πραγματοποίηση και ενσωμάτωση των αλλαγών στο υπό ανάπτυξη σύστημα.

- Οι αλλαγές των απαιτήσεων πρέπει να παρέχονται με σαφήνεια και πληρότητα, κατά το δυνατόν, στις αντίστοιχες μικτές ομάδες ανάπτυξης (κατασκευαστών και χρηστών) ανά λειτουργική μονάδα του οργανισμού και σε όλους τους διαφορετικούς γεωγραφικούς τόπους που είναι εγκατεστημένοι, προκειμένου οι ομάδες αυτές να λογοδοτούν αν, όπου και όταν απαιτείται.

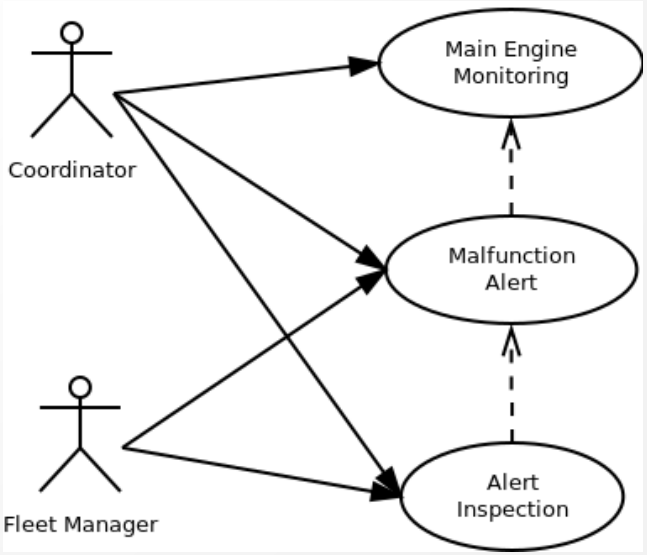
Λογική Ενοποιημένη Διεργασία: Χαρακτηριστικά

Τα βασικά χαρακτηριστικά της μεθοδολογίας RUP είναι, σύμφωνα με τις βασικές αρχές της αντικειμενοστρεφούς μεθοδολογίας, τα ακόλουθα:

- Είναι καθοδηγούμενη από περιπτώσεις χρήσης *(use-case driven),*
- Είναι αρχιτεκτονικο-κεντρική *(architecture centric),* και
- Είναι επαναληπτική και επαυξητική *(iterative & incremental)*.

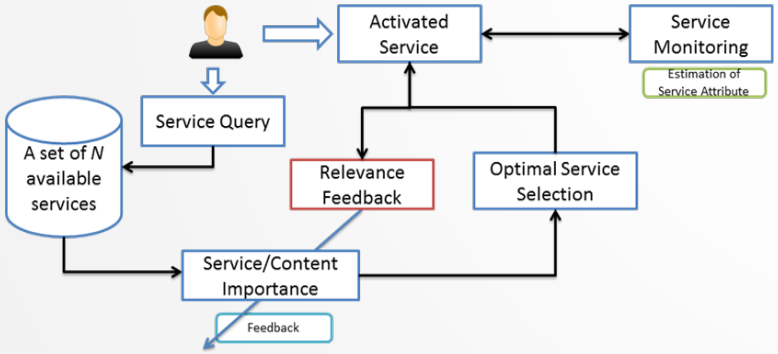
Λογική Ενοποιημένη Διεργασία: Χαρακτηριστικά

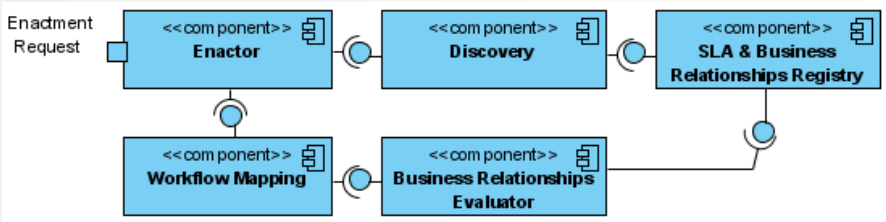
i. **Καθοδηγούμενη από περιπτώσεις χρήσης** – Κατασκευάζονται σενάρια από τα οποία αντλούνται και συνάγονται οι λειτουργικές και τεχνικές απαιτήσεις του συστήματος. Τα σενάρια αυτά κατασκευάζονται χρησιμοποιώντας την τεχνική των διαγραμμάτων περιπτώσεων χρήσης *(use-case diagrams)* της UML και μπορεί να περιγράφουν επιθυμητές ή/και ανεπιθύμητες ακολουθίες γεγονότων. Οι περιπτώσεις χρήσης χρησιμοποιούνται ως βασικά στοιχεία για τον καθορισμό της συμπεριφοράς, την επαλήθευση της αρχιτεκτονικής, τον έλεγχο και την επικοινωνία μεταξύ των εμπλεκομένων στην ανάπτυξη του συστήματος.

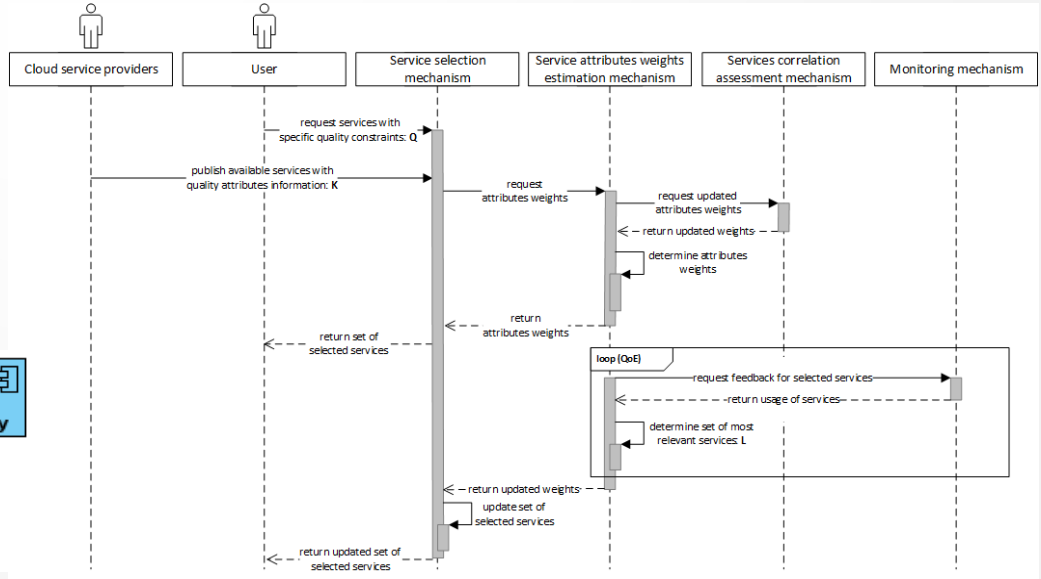


Λογική Ενοποιημένη Διεργασία: Χαρακτηριστικά

ii. **Αρχιτεκτονικο-κεντρική** – Ακολουθείται μια διεργασία ανάπτυξης συστημάτων που βασίζεται στην αρχιτεκτονική του συστήματος. Η αρχιτεκτονική του συστήματος χρησιμοποιείται ως βασικό μέσο για την κατανόηση, την κατασκευή, τη διαχείριση και την εξέλιξη του υπό κατασκευή συστήματος οδηγώντας στην προδιαγραφή *(specification)*, στην κατασκευή *(construction)* και στην τεκμηρίωση *(documentation)* αυτού. Για το σκοπούς αυτούς πρέπει να υποστηρίζονται τουλάχιστον τρεις **διαφορετικές όψεις της αρχιτεκτονικής**: η λειτουργική *(functional - conceptual)*, η δομική ή στατική *(structural or static = component model)* και η δυναμική ή συμπεριφορική *(dynamic or behavioral)* με αντίστοιχες διαγραμματικές τεχνικές που περιλαμβάνονται στην UML.







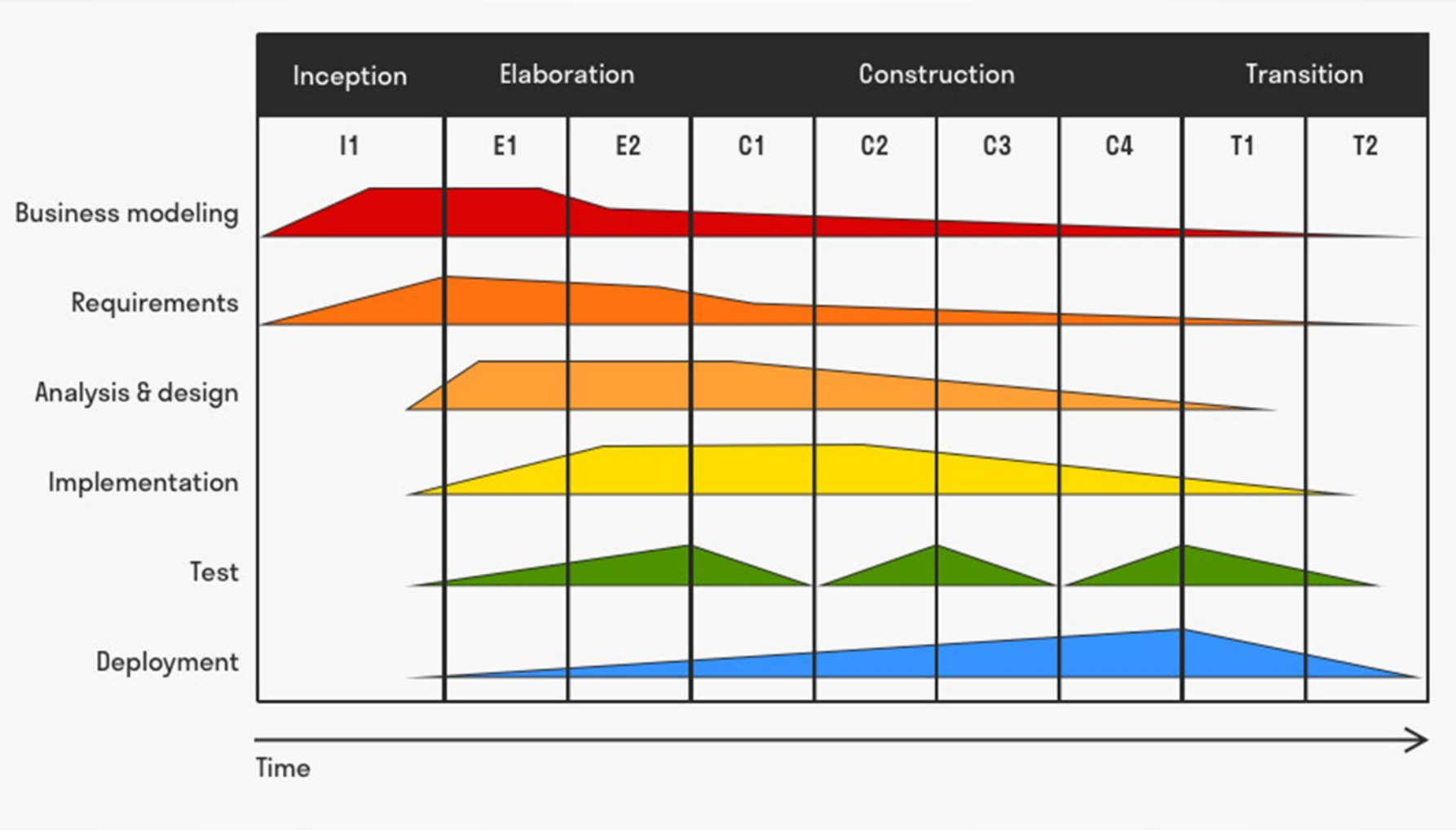
Λογική Ενοποιημένη Διεργασία: Χαρακτηριστικά

iii. **Επαναληπτική και επαυξητική**: Η ακολουθούμενη **διεργασία** ανάπτυξης συστημάτων είναι **επαναληπτική** και **επαυξητική**.

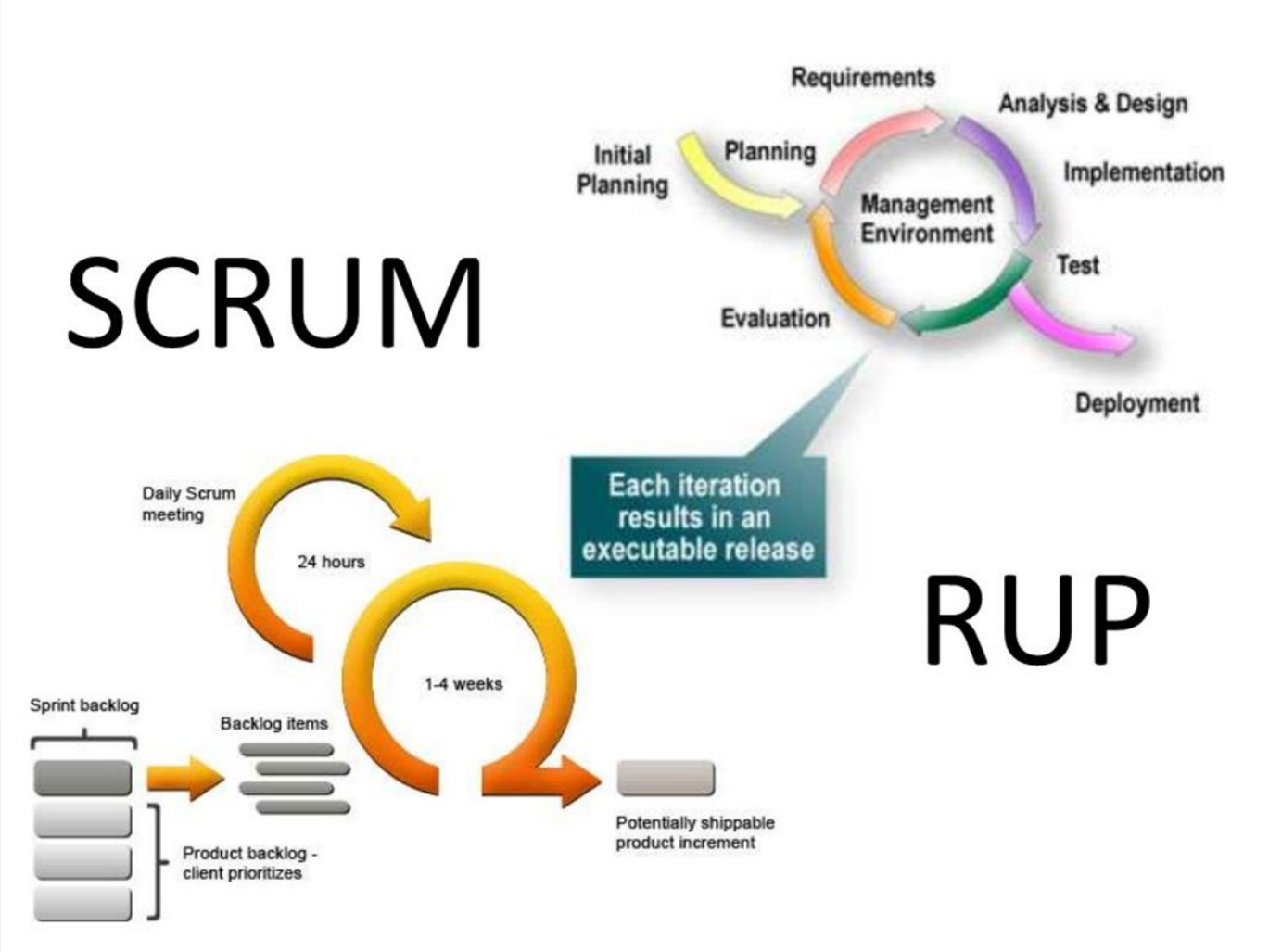
- Η επαναληπτική διεργασία αφορά στη διαχείριση των εκτελέσιμων εκδόσεων ενός συστήματος και
- η επαυξητική διεργασία αφορά στη συνεχή ολοκλήρωση της αρχιτεκτονικής για την παραγωγή των εκτελέσιμων εκδόσεων του συστήματος.

Το σύστημα κατασκευάζεται κατά διαδοχικές εκδόσεις, κάθε μία από τις οποίες διαθέτει βελτιωμένα χαρακτηριστικά, και δεν παρουσιάζεται ως τετελεσμένο στο χρονικό πέρας του έργου.

Scrum or RUP


Inception
Elaboration
Construction
Transition
I1
E1
E2
C1
C2
C3
C4
T1
T2
Business modeling
Requirements
Analysis & design
Implementation
Test
Deployment
Time

Scrum or RUP


SCRUM
Daily Scrum meeting
24 hours
1-4 weeks
Sprint backlog
Backlog items
Product backlog - client prioritizes
Potentially shippable product increment
Each iteration results in an executable release
RUP
Requirements
Analysis & Design
Implementation
Test
Deployment
Evaluation
Initial Planning
Planning
Management Environment

(Key) Differences

- Scope of the project
  - RUP: Specific (inception phase) -> will not change
  - SCRUM: Flexible -> might change
- Planning of the project
  - RUP: Well-specified (milestones) with iterations
  - SCRUM: Much more dynamic (product owner decide when something is completed) with iterations
- Cycle
  - RUP: 4 phases with various overlapping processes (e.g. requirements, analysis, implementation, etc) for potentially different components
  - SCRUM: Each cycle is “atomic” / “complete” per component

Περιεχόμενα

- Σύνοψη προηγούμενης διάλεξης
- Λογική Ενοποιημένη Διεργασία
- Ανάπτυξη συστημάτων συνιστωσών λογισμικού
  - Version Control
  - Σύνθεση των ΠΣ που βασίζονται σε συνιστώσες
  - Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών
- Υπηρεσιοστρεφής ανάπτυξη συστημάτων
  - Η χρήση των υπηρεσιών ιστού
- DevOps

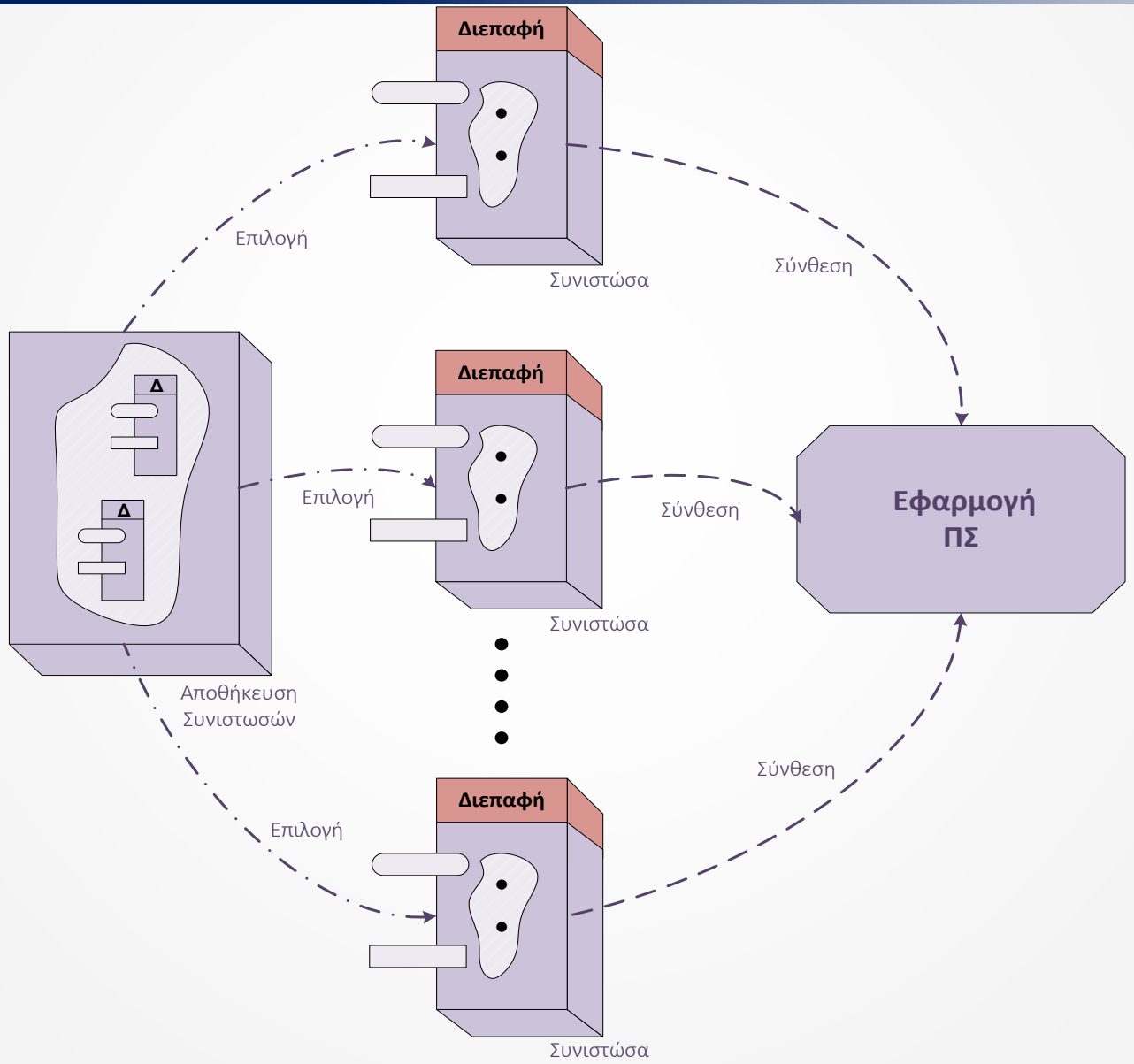
Ανάπτυξη συστημάτων συνιστωσών λογισμικού

Η ανάπτυξη συστημάτων λογισμικού και, κατά συνέπεια, πληροφοριακών συστημάτων, διέπεται από μερικές θεμελιώδεις αρχές επί των οποίων υπάρχει γενική συμφωνία μεταξύ των ασχολούμενων με τα συναφή επιστημονικά πεδία:

- **Ανεξάρτητη ανάπτυξη λογισμικού**: Τα μεγάλα και πολύπλοκα ΠΣ αποτελούνται συχνά από (επιθυμητά διαλειτουργικές) συνιστώσες που έχουν αναπτυχθεί ανεξάρτητα από διαφορετικούς κατασκευαστές. Για τη διευκόλυνση της ανεξάρτητης ανάπτυξης των συνιστωσών των συστημάτων, είναι σημαντικό (α) να αποσυνδεθεί η κατασκευή τους από τις συγκεκριμένες χρήσεις τους και (β) να παρέχονται διαλειτουργικές διεπαφές που βασίζονται σε προδιαγραφές της συμπεριφοράς των συνιστωσών οι οποίες είναι αφηρημένες και ουδέτερες.

- **Επαναχρησιμοποιησιμότητα:** Βασικά, ένα μεγάλο και πολύπλοκο ΠΣ θα αποτελείται από διάφορα λογικά μέρη κάποια από τα οποία θα είναι λογισμικό ειδικού σκοπού ενώ κάποια άλλα θα συνιστούν συνιστώσες οι οποίες αποτελούνται από ειδικά σχεδιασμένες και διασυνδεδεμένες προ-υπάρχουσες, μικρότερες συνιστώσες (ενδο- ή δια-τομεακές/τμηματικές).
- **Ποιότητα λογισμικού:** Είναι αναγκαίο για μία συνιστώσα ή ένα σύστημα να επιδεικνύει την επιθυμητή συμπεριφορά, όταν ελέγχεται (δοκιμάζεται) είτε μέσω λογικών συλλογισμών είτε μέσω εντοπισμού λειτουργικών και άλλων σφαλμάτων. Η προσέγγιση του ελέγχου της ποιότητας λογισμικού πρέπει να είναι σπονδυλωτή και επεκτάσιμη.
- **Συντηρησιμότητα:** Κάθε ΠΣ πρέπει να είναι καταληπτό και εύκολα αναβαθμίσιμο και εξελίξιμο.

Ανάπτυξη εφαρμογών ΠΣ από προκατασκευασμένες συνιστώσες λογισμικού



Βασικές έννοιες και ορισμοί

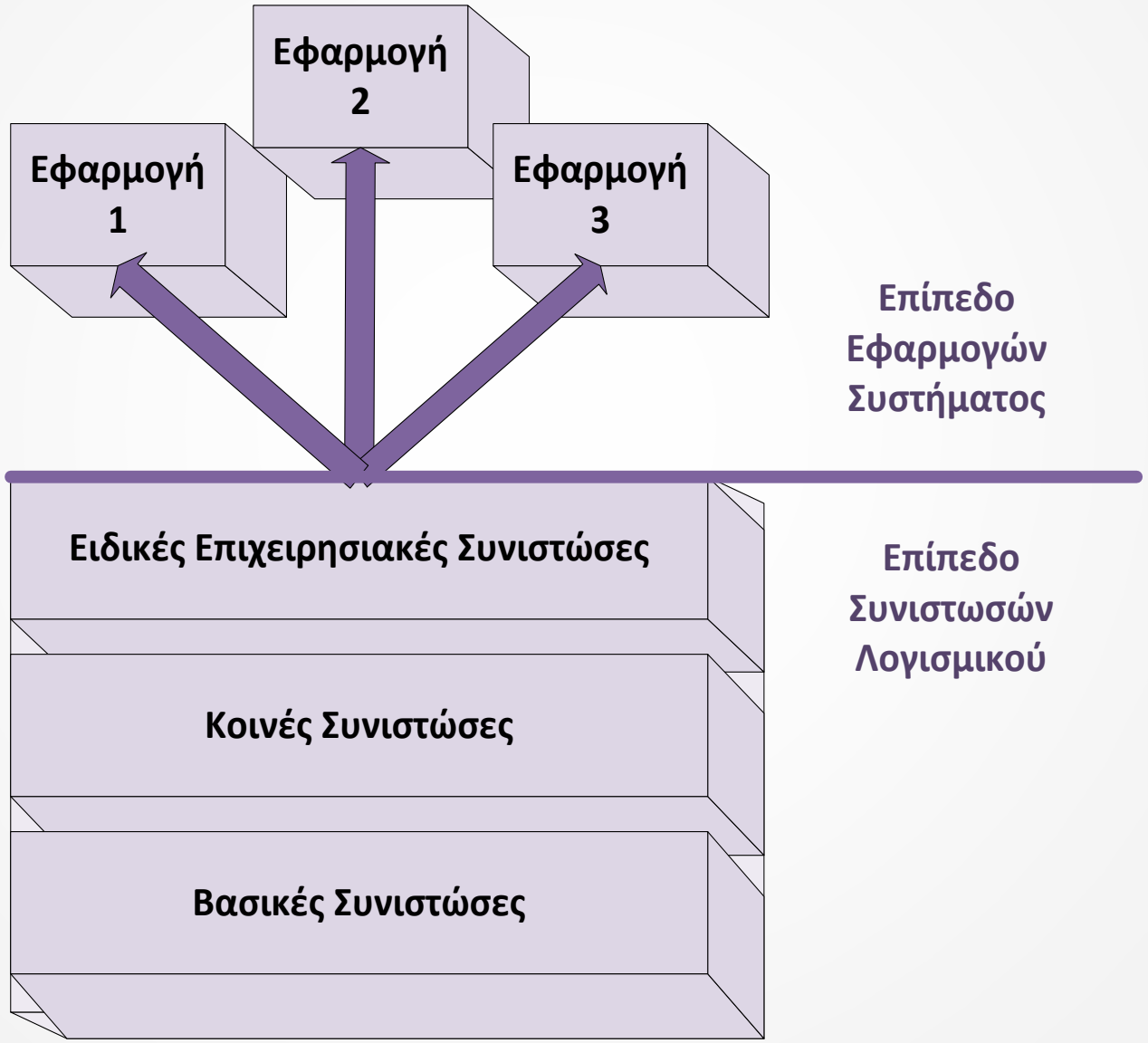
Τα βασικά (και ευκταία) χαρακτηριστικά της ανάπτυξης ΠΣ κατά συνιστώσες είναι:

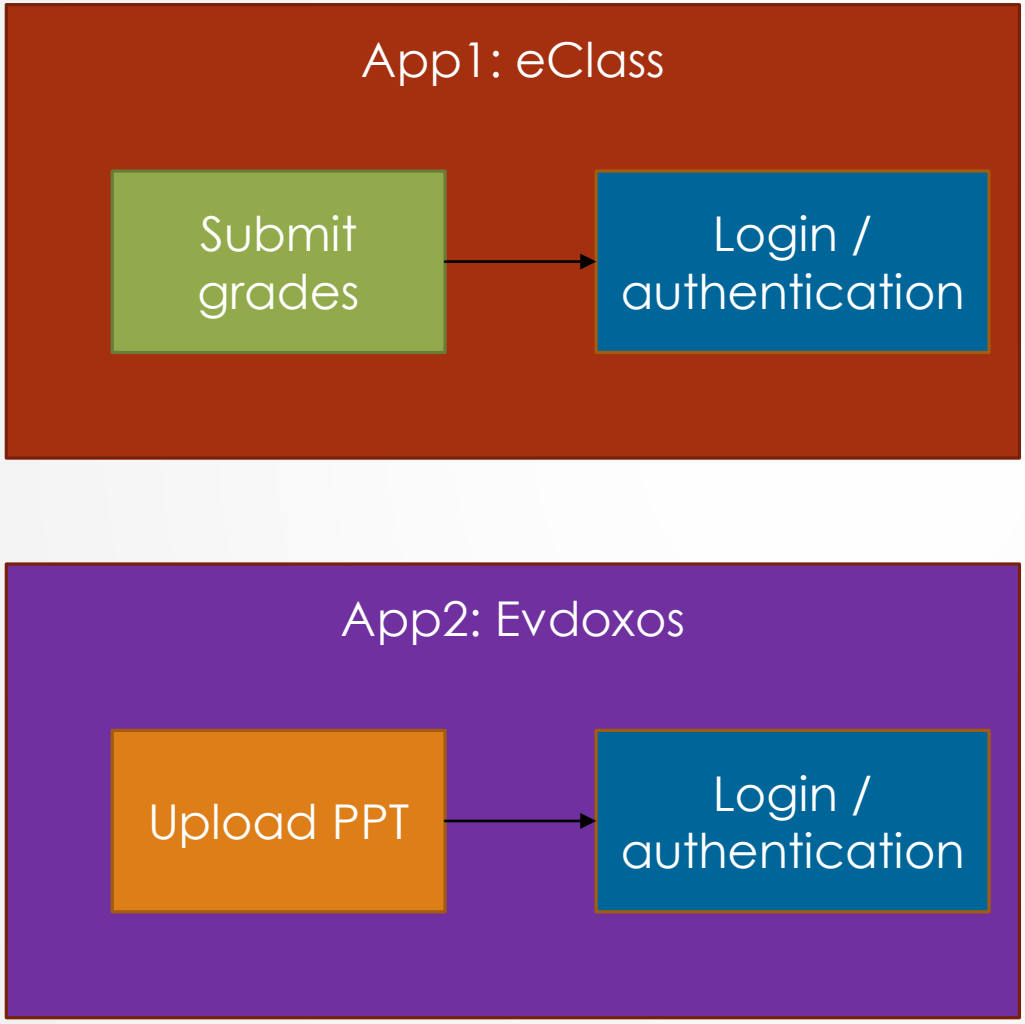
- Ότι **οι συνιστώσες πρέπει να είναι ανεξάρτητες μεταξύ τους** και να **προσδιορίζονται μόνο μέσω των διεπαφών τους**.
- Ότι οι συνιστώσες πρέπει να βασίζονται σε διεθνή **πρότυπα** προκειμένου να διευκολύνεται η ολοκλήρωση/ διαλειτουργικότητα μεταξύ τους.
- Ότι η ανάπτυξη ΠΣ κατά συνιστώσες συνιστά μια διεργασία ανάπτυξης συστημάτων λογισμικού που αναφέρεται ευθέως στην ιδέα της **επαναχρησιμοποίησης του υπάρχοντος λογισμικού**.

Βασικές έννοιες και ορισμοί

- Κάθε συνιστώσα παρέχει μια υπηρεσία που είναι **ανεξάρτητη από την πλατφόρμα λογισμικού** στην οποία μπορεί να εκτελεστεί ή από την **γλώσσα προγραμματισμού** στην οποία έχει γραφεί.
- Οι συνιστώσες είναι ανεξάρτητες εκτελέσιμες οντότητες οι οποίες μπορεί να έχουν σχηματιστεί από τη σύνθεση ενός ή περισσοτέρων εκτελέσιμων αντικειμένων ή από τη σύνθεση άλλων συνιστωσών.
- Οι διεπαφές των συνιστωσών “δημοσιεύονται” έτσι ώστε όλες οι αλληλεπιδράσεις των χρηστών με τις συνιστώσες να πραγματοποιούνται μέσω των δημοσιευμένων διεπαφών τους.

Βασικές έννοιες και ορισμοί



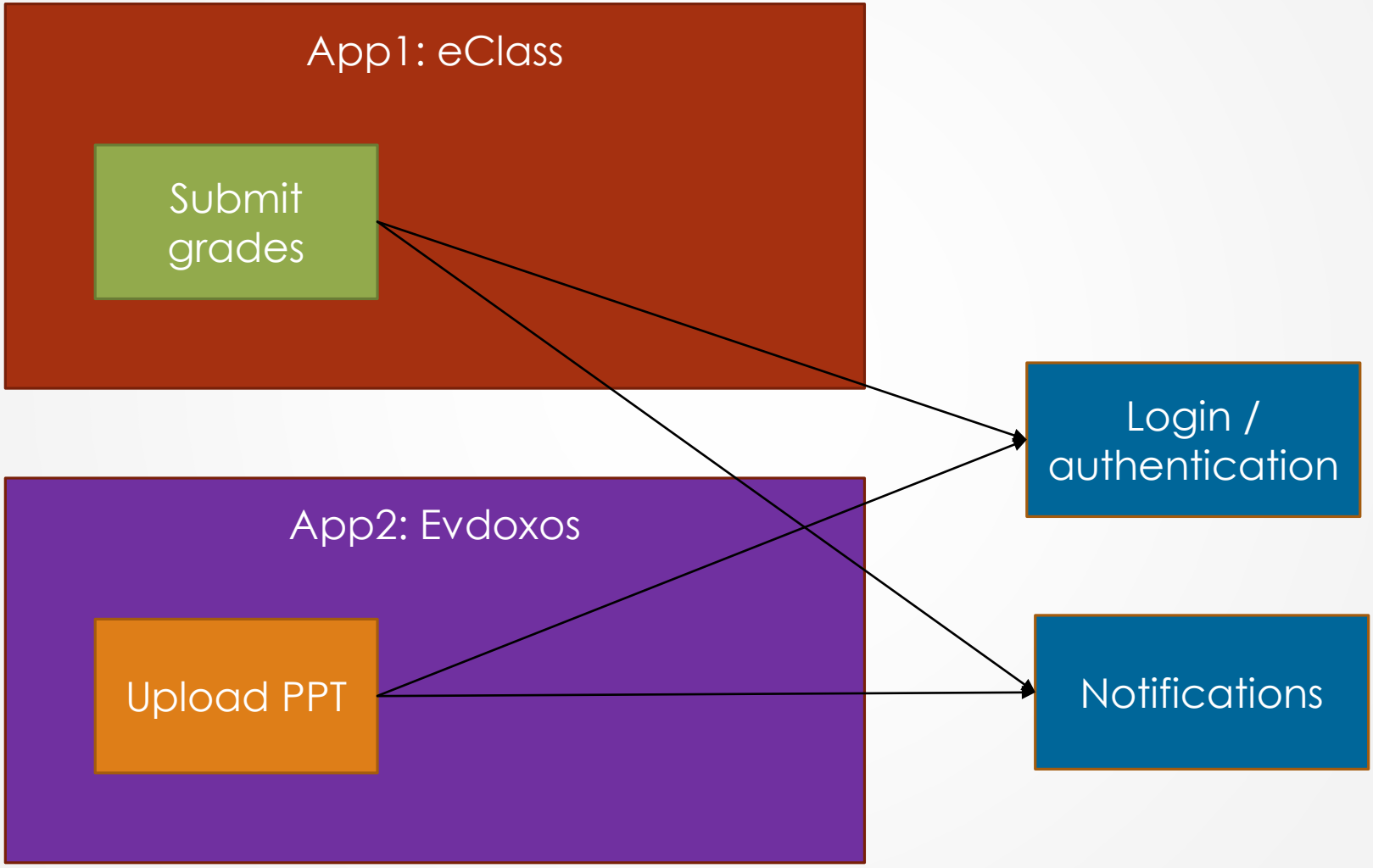


Business component: Submit grades

Business component: Upload PPT

Common component: Login / authentication

COMPONENTS CODE & DEPLOYMENT: 1 for each common component (reusability & maintenance)



Βασικές έννοιες και ορισμοί

Το κύριο επίπεδο των συνιστωσών λογισμικού διακρίνεται σε τρία επιμέρους επίπεδα:

- Το άνω επιμέρους επίπεδο αποτελείται από συνιστώσες που αναφέρονται σε ένα συγκεκριμένο επιχειρησιακό πεδίο ή πεδίο εφαρμογών, συμπεριλαμβανομένων των συνιστωσών που μπορούν να χρησιμοποιηθούν σε περισσότερες από μία εφαρμογές.
- Το ενδιάμεσο επιμέρους επίπεδο αποτελείται από ενδιάμεσο λογισμικό δι-επιχειρησιακών συνιστωσών που συντίθεται μέσω κοινού λογισμικού και διεπαφών σε άλλες συγκεκριμένες οντότητες.
- Τέλος, το κάτω επιμέρους επίπεδο περιλαμβάνει βασικές συνιστώσες που αλληλεπιδρούν με τα υποκείμενα λειτουργικά συστήματα και το υλικό.

Ανάπτυξη - αποθετήρια

- Από έναν αποθηκευτικό χώρο στον οποίο αποθηκεύονται συνιστώσες λογισμικού επιλέγονται οι κατάλληλες συνιστώσες προκειμένου να διασυνδεθούν και να σχηματιστούν επιμέρους εφαρμογές κάθε συστήματος.
- Οι προκατασκευασμένες συνιστώσες λογισμικού είναι διαθέσιμες σε μία ή περισσότερες εφαρμογές ενός ή περισσοτέρων ΠΣ.

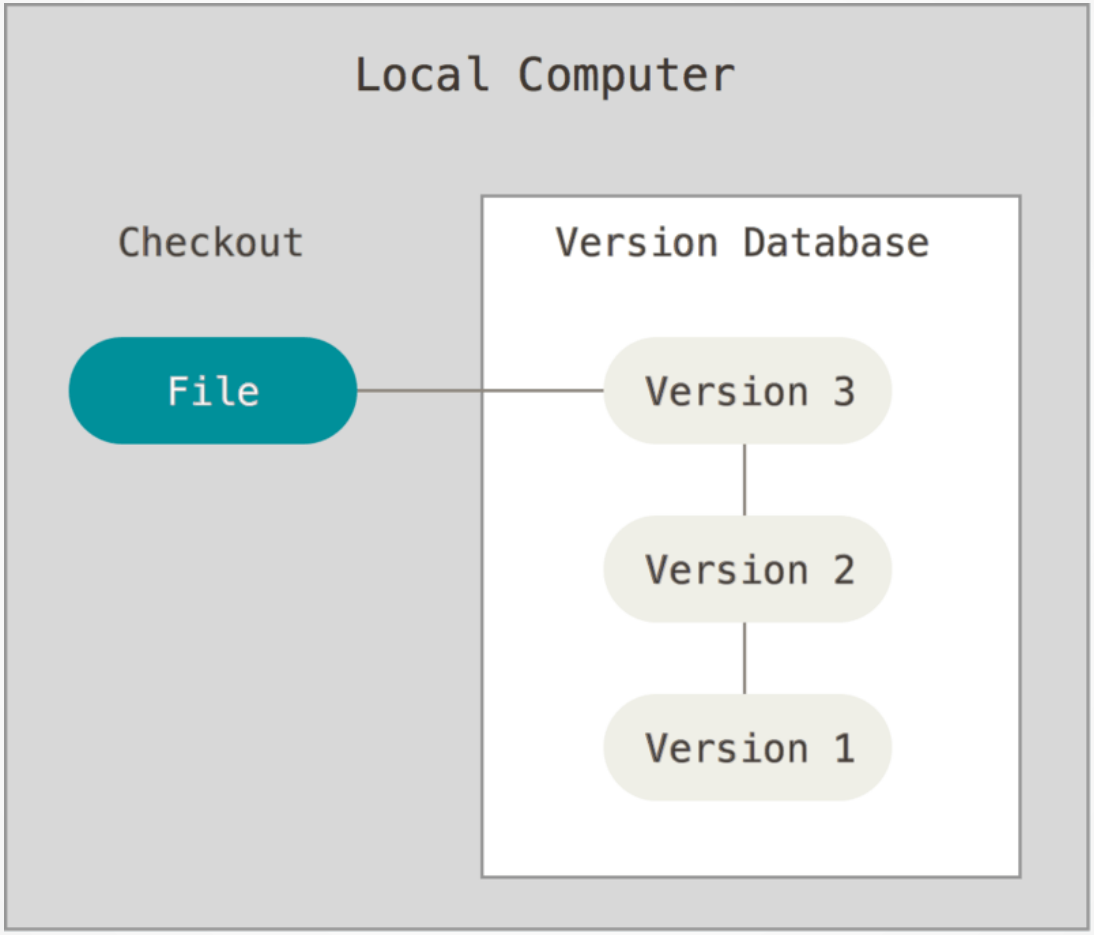
Περιεχόμενα

- Σύνοψη προηγούμενης διάλεξης
- Λογική Ενοποιημένη Διεργασία
- Ανάπτυξη συστημάτων συνιστωσών λογισμικού
  - Version Control
  - Σύνθεση των ΠΣ που βασίζονται σε συνιστώσες
  - Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών
- Υπηρεσιοστρεφής ανάπτυξη συστημάτων
  - Η χρήση των υπηρεσιών ιστού
- DevOps

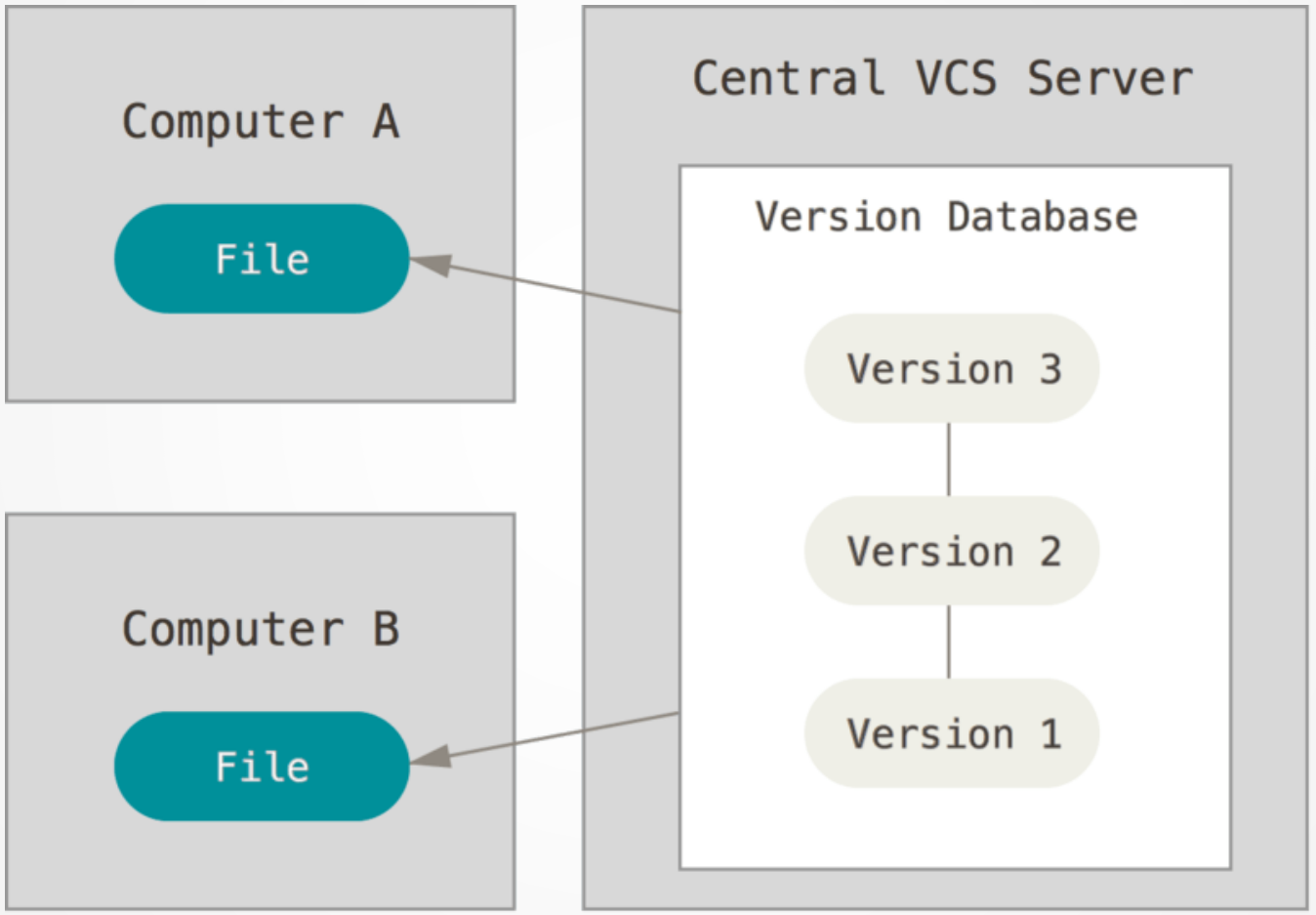
Version Control

- Version control: Σύστημα καταγραφής αλλαγών σε 1 ή περισσότερα αρχεία ώστε να είναι εφικτή η μελλοντική ανάκτηση εκδόσεων
- Τύποι
  - Local
  - Centralized
  - Distributed
- Παραδείγματα
  - CVS
  - Subversion
  - Git

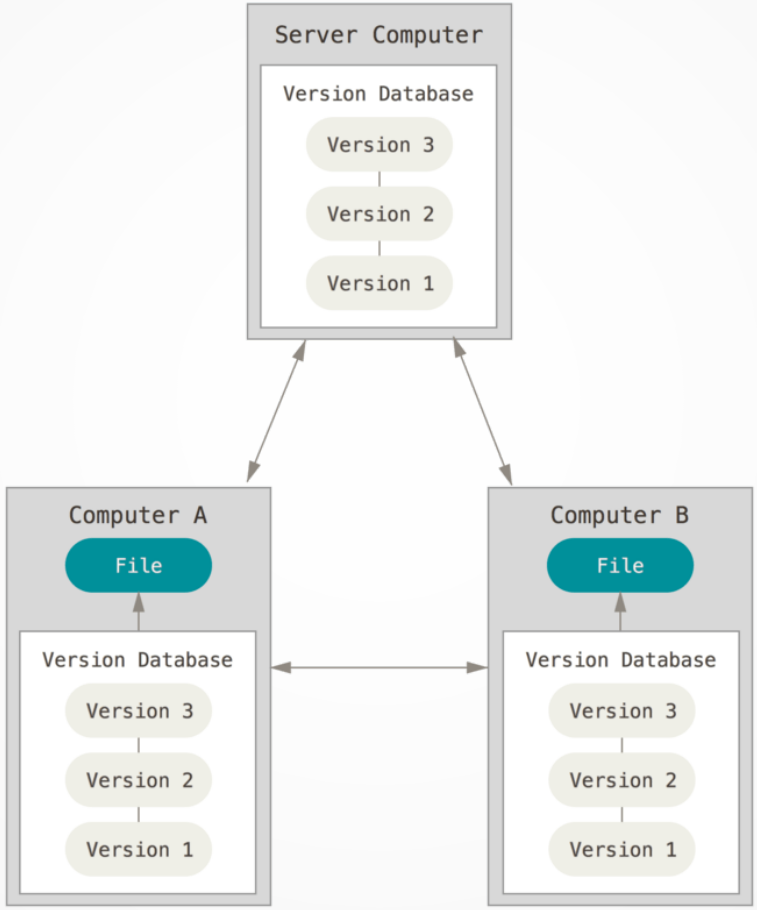
Local Version Control Systems



Centralized Version Control Systems + LOCKING!



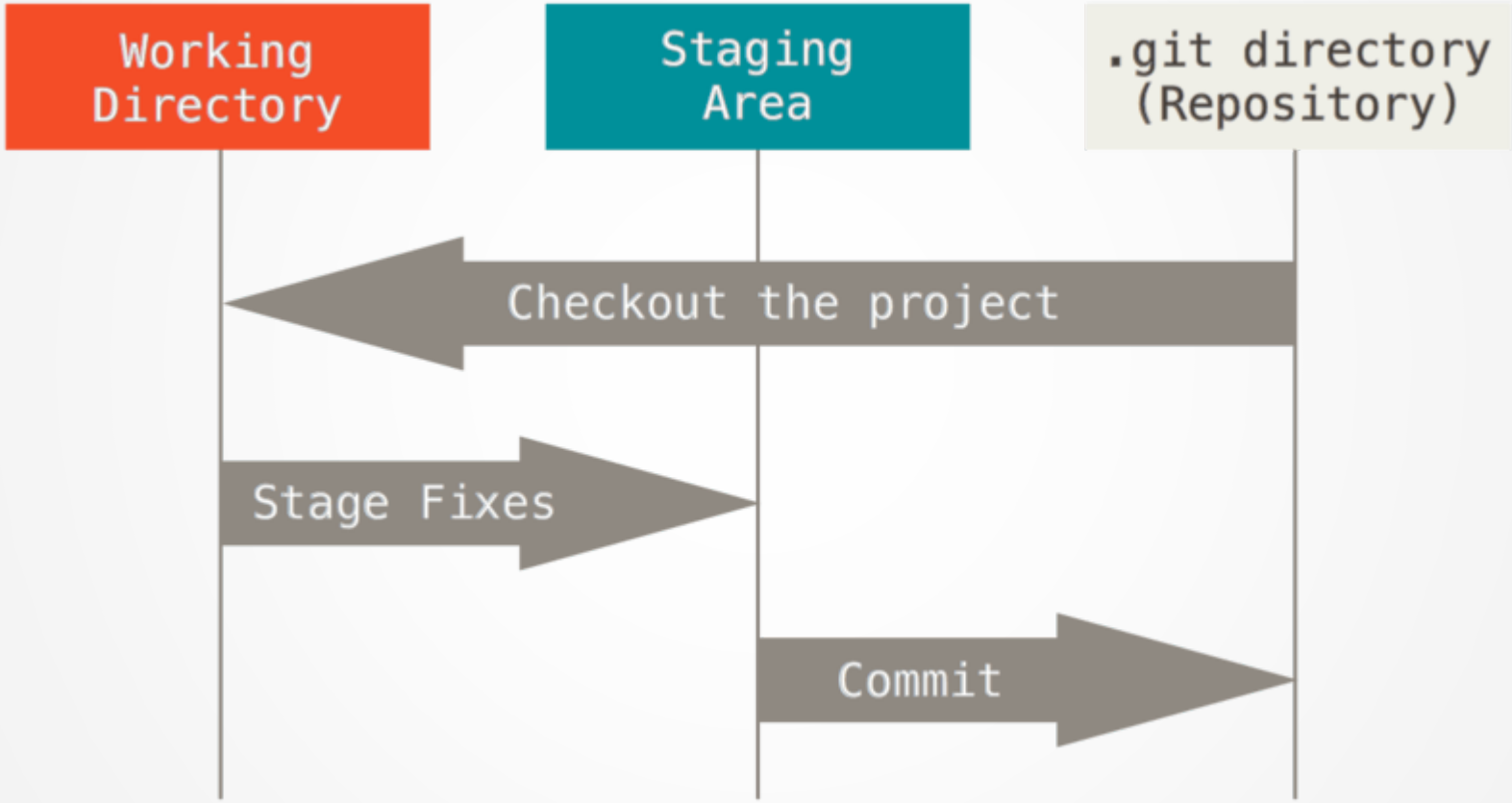
Distributed Version Control Systems



Ιστορία

- Ξεκίνησε η υλοποίηση τον Απρίλιο 2005, καθώς πολλοί προγραμματιστές του Linux kernel σταμάτησαν να χρησιμοποιούν το BitKeeper (proprietary source-control management - SCM)
- Timeline
  - Υλοποίηση ξεκίνησε 3 Απριλίου 2005
  - Ο Torvalds ανακοίνωσε στις 6 Απριλίου ότι θα είναι self-hosting στις 7 Απριλίου
  - Το πρώτο merge πολλαπλών branches έγινε στις 18 Απριλίου
  - Στις 29 Απριλίου το Git χρησιμοποιούνταν για τα patches του Linux kernel (6.7 patches / second)
  - Στις 16 Ιουνίου το Git διαχειριζόταν την έκδοση του kernel (2.6.12)
- Όνομα git (σημαίνει unpleasant person in British English slang): "I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'."

Git States & Git repositories for developers (i.e. development phase)



Docker (and containers and Docker Hubs / repositories) are for administrators (i.e. operations phase)

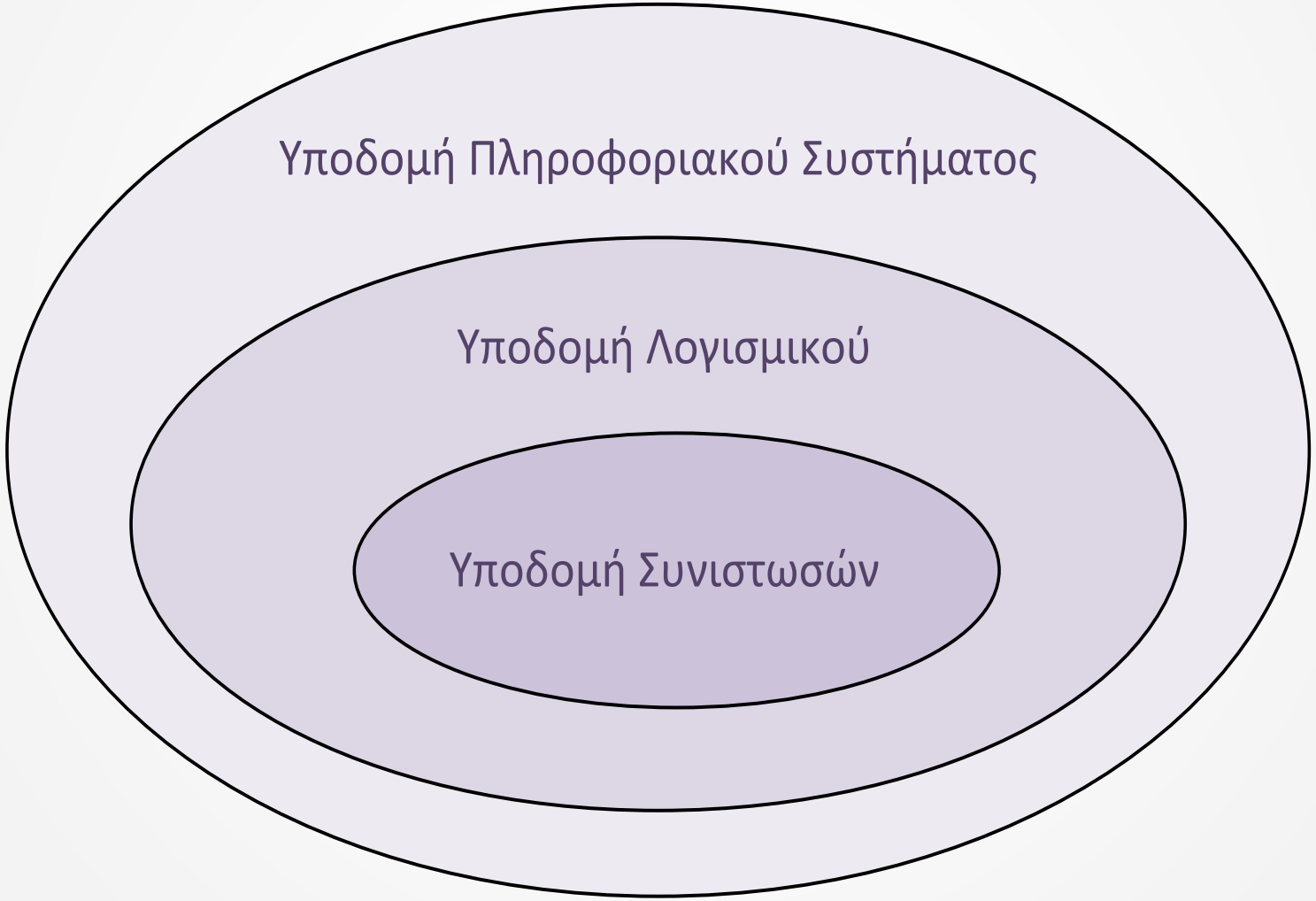
Περιεχόμενα

- Σύνοψη προηγούμενης διάλεξης
- Λογική Ενοποιημένη Διεργασία
- Ανάπτυξη συστημάτων συνιστωσών λογισμικού
  - Version Control
  - Σύνθεση των ΠΣ που βασίζονται σε συνιστώσες
  - Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών
- Υπηρεσιοστρεφής ανάπτυξη συστημάτων
  - Η χρήση των υπηρεσιών ιστού
- DevOps

Η σύνθεση των ΠΣ που βασίζονται σε συνιστώσες

- Γενικά, ένα ΠΣ που βασίζεται σε συνιστώσες λογισμικού είναι σημαντικά πιο περίπλοκο από ένα παραδοσιακό ΠΣ που βασίζεται σε μονολιθικό λογισμικό.
- Κατά μία προσέγγιση, ένα τέτοιο σύστημα μπορεί να θεωρηθεί ότι αποτελείται από τρία επίπεδα:
  - Το **εσωτερικό επίπεδο** συνιστά την υποδομή συνιστωσών (δηλαδή, τις επιμέρους συνιστώσες μαζί με το αναγκαίο "συγκολλητικό" λογισμικό που τις καταστεί διαλειτουργικές).
  - Το **ενδιάμεσο επίπεδο** συνιστά την υποδομή λογισμικού, ή την υποδομή εφαρμογών (δηλαδή, την ομαδοποίηση συνεργαζόμενων συνιστωσών σε εφαρμογές λογισμικού)
  - Το **εξωτερικό επίπεδο** συνιστά την υποδομή του ΠΣ (δηλαδή, το ΠΣ που υποστηρίζεται ή αποτελείται από τις διάφορες εφαρμογές).

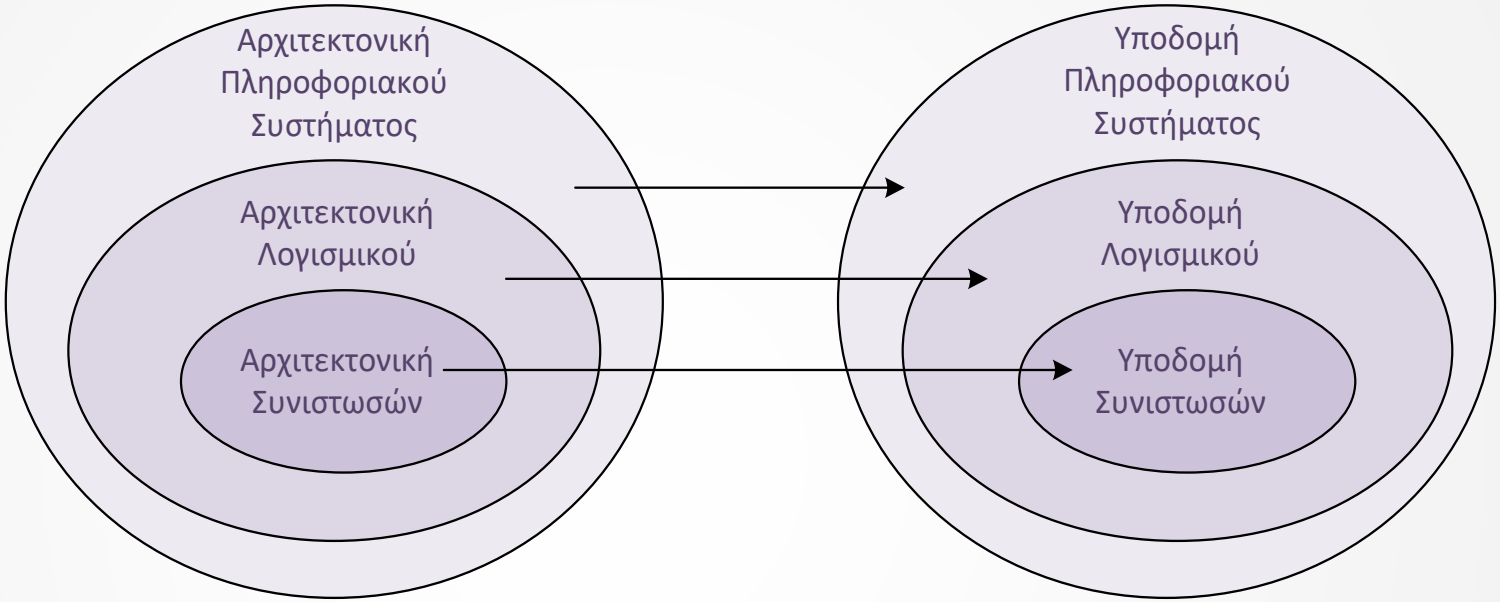
Υποδομές ΠΣ που βασίζονται σε συνιστώσες λογισμικού

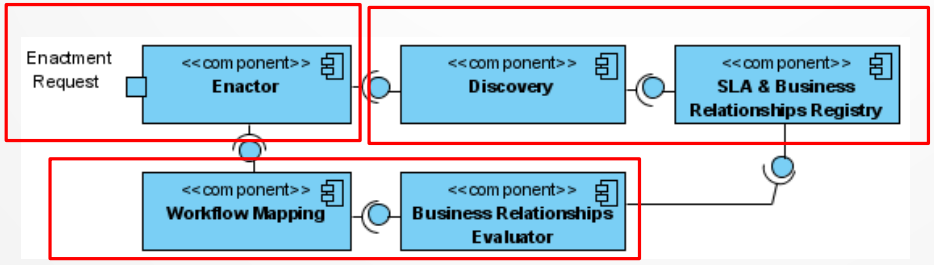


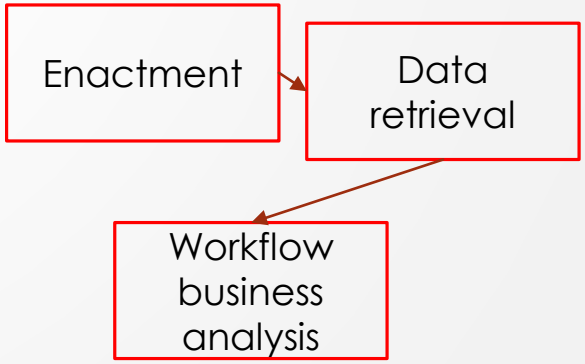
Η σύνθεση των ΠΣ που βασίζονται σε συνιστώσες

- Κάθε ένα από τα τρία επίπεδα πρέπει να εκφράζεται από (ή να αντιστοιχίζεται σε) μια αρχιτεκτονική που καθορίζει την τεχνική υποδομή.
- Αυτό σημαίνει ότι κάθε μία από τις τρεις υποδομές πραγματώνεται μέσω της αντίστοιχης τεχνικής αρχιτεκτονικής που, με τη σειρά του, συνεπάγεται ότι αν χρησιμοποιηθεί μια άτυπη αρχιτεκτονική τότε θα παραχθεί μια άτυπη υποδομή.
- Έτσι, σε κάθε ένα από τα επίπεδα που περιγράφηκαν παραπάνω αντιστοιχεί μια αρχιτεκτονική, όπως φαίνεται στο ακόλουθο σχήμα.

Οι διάφορες αρχιτεκτονικές και οι αντίστοιχές τους υποδομές ΠΣ που βασίζονται σε συνιστώσες λογισμικού







Η σύνθεση των ΠΣ που βασίζονται σε συνιστώσες

- Κάθε αρχιτεκτονική ορίζει τις συνιστώσες ενός συστήματος και την συλλειτουργία τους καθώς και την βασική δομή και τον βασικό σχεδιασμό.
- Οι παραπάνω αρχιτεκτονικές περιγράφονται ως ακολούθως:
  - **Αρχιτεκτονική ΠΣ:** Η αρχιτεκτονική ΠΣ ορίζει πώς η επιχείρηση κατανέμει τις πληροφορίες και την επεξεργασία πληροφοριών σε διάφορα επιμέρους υποσυστήματα, οριοθετώντας τα με αυτόν τον τρόπο.
  - **Αρχιτεκτονική Λογισμικού:** Η αρχιτεκτονική λογισμικού ορίζει τα επιμέρους υποσυστήματα λογισμικού που συνιστούν τα δομικά συστατικά του συστήματος καθώς και τις διασυνδέσεις τους μέσω συγκεκριμένων διεπαφών, ορίζοντας με αυτόν τον τρόπο σε σημαντικό βαθμό πώς αλληλεπιδρούν οι κόμβοι που αντιστοιχούν σ’ αυτά τα υποσυστήματα.

Η σύνθεση των ΠΣ που βασίζονται σε συνιστώσες

- **Αρχιτεκτονική Συνιστωσών:** Η αρχιτεκτονική συνιστωσών καθορίζει τις επιμέρους πλατφόρμες λογισμικού για την ανάπτυξη και λειτουργία των συνιστωσών και τον τρόπο επίτευξης της διαλειτουργικότητας μεταξύ των συνιστωσών.

Περιεχόμενα

- Σύνοψη προηγούμενης διάλεξης
- Λογική Ενοποιημένη Διεργασία
- Ανάπτυξη συστημάτων συνιστωσών λογισμικού
  - Version Control
  - Σύνθεση των ΠΣ που βασίζονται σε συνιστώσες
  - Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών
- Υπηρεσιοστρεφής ανάπτυξη συστημάτων
  - Η χρήση των υπηρεσιών ιστού
- DevOps

Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών

- Όπως αναφέρθηκε προηγουμένως, στα συστήματα συνιστωσών διακρίνονται τρία διαφορετικά επίπεδα:
  - Το επίπεδο του πληροφοριακού συστήματος
  - Το επίπεδο των εφαρμογών, και
  - Το επίπεδο των συνιστωσών
- Η διάρθρωση αυτή υποδηλώνει ότι ένα ΠΣ συνιστωσών αποτελείται από εφαρμογές κάθε μία από τις οποίες αποτελείται από συνιστώσες.

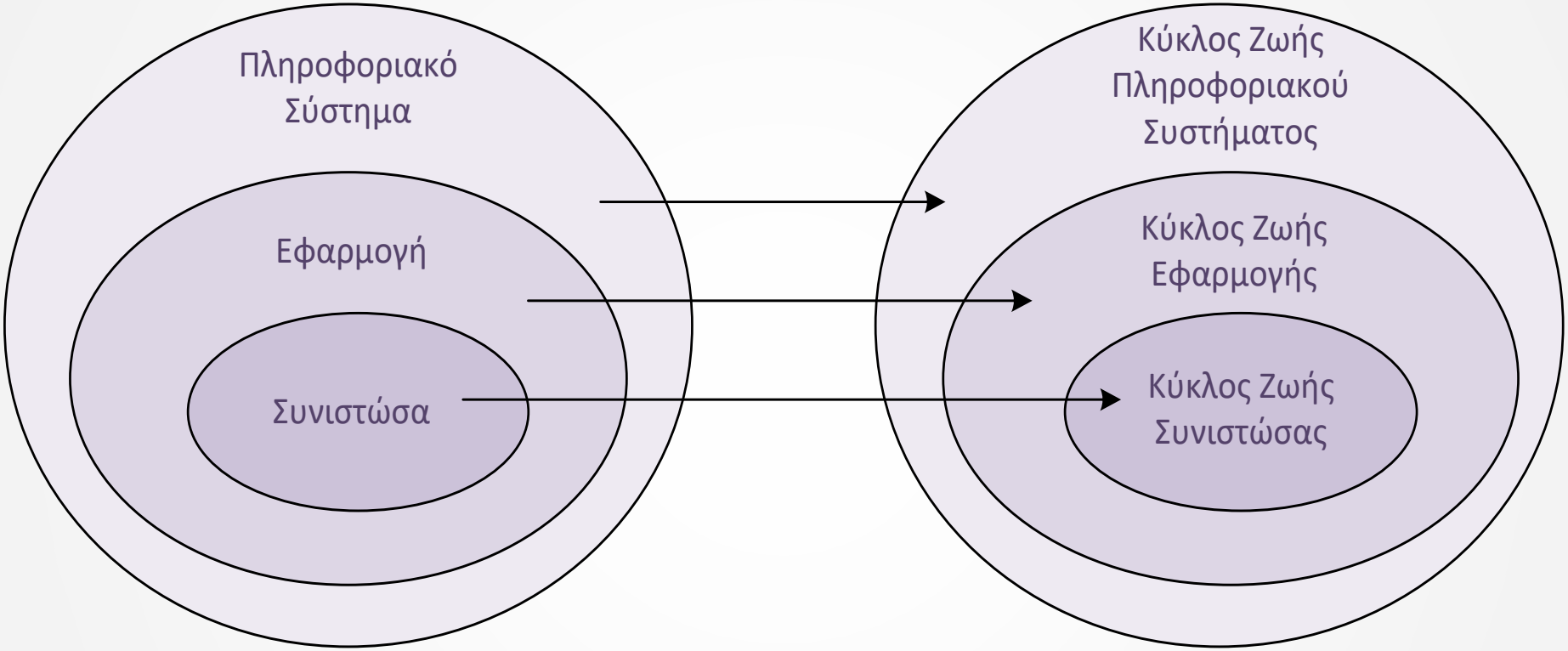
Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών

- Κατ' ακολουθία, στο πλαίσιο της ανάπτυξης ΠΣ συνιστωσών ορίζονται και χρησιμοποιούνται τρεις διαφορετικοί κύκλοι ζωής για τα τρία επίπεδα ενός ΠΣ συνιστωσών, αντίστοιχα, αντί να ορίζεται και να χρησιμοποιείται ένα μοντέλο κύκλου ζωής για την ανάπτυξη ολόκληρου του ΠΣ.
- Συνεπώς, ορίζονται και χρησιμοποιούνται οι ακόλουθοι κύκλοι ζωής:
  - Ένας κύκλος ζωής για κάθε συνιστώσα,
  - Ένας κύκλος ζωής για κάθε εφαρμογή, και
  - Ένας κύκλος ζωής για το ΠΣ.

Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών

- Χρησιμοποιώντας τρία διαφορετικά μοντέλα κύκλου ζωής επιτυγχάνεται μια πιο ώριμη και πιο καθαρή περιγραφή της ανάπτυξης και της συντήρησης πληροφοριακών συστημάτων συνιστωσών καθότι μπορεί να εστιάσει κανείς σε μία συνιστώσα ή σε ολόκληρο το ΠΣ.
- Υπάρχουν διαφορετικά μοντέλα που περιγράφουν την ανάπτυξη και συντήρηση του αντιστοίχου επιπέδου ξεχωριστά και, ταυτόχρονα, περιγράφονται οι εξαρτήσεις τους.

Οι τρεις κύκλοι ζωής της ανάπτυξης συστημάτων συνιστωσών



Περιεχόμενα

- Σύνοψη προηγούμενης διάλεξης
- Λογική Ενοποιημένη Διεργασία
- Ανάπτυξη συστημάτων συνιστωσών λογισμικού
  - Version Control
  - Σύνθεση των ΠΣ που βασίζονται σε συνιστώσες
  - Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών
- Υπηρεσιοστρεφής ανάπτυξη συστημάτων
  - Η χρήση των υπηρεσιών ιστού
- DevOps

Υπηρεσίες - Ορισμός

- Ως **“υπηρεσία”** ορίζεται μια λογική αναπαράσταση μιας επιχειρησιακής δραστηριότητας που επαναλαμβάνεται και παράγει συγκεκριμένα αποτελέσματα (π.χ. έλεγχος του πιστωτικού υπολοίπου ενός πελάτη τράπεζας, παραγωγή μηνιαίων αναφορών πωλήσεων).
  - Είναι αυτόνομη και αυτοδύναμη
  - Μπορεί να αποτελείται από άλλες υπηρεσίες
  - Είναι “μαύρο κουτί” για τους καταναλωτές της υπηρεσίας
- Αλλιώς, **“υπηρεσία”** είναι μια λειτουργία ή επιχειρησιακή διεργασία ενός οργανισμού η οποία είναι καλά ορισμένη, είναι αυτοδύναμη και δεν εξαρτάται από το περιβάλλον ή την κατάσταση άλλων υπηρεσιών.

Υπηρεσίες - Ορισμός

- Ως **“υπηρεσιοστρεφής προσανατολισμός”** ορίζεται ένας τρόπος σκέψης που βασίζεται στην θεώρηση ότι ο υπό μελέτη κόσμος αποτελείται από υπηρεσίες, υπηρεσιοστρεφή ανάπτυξη και εκροές από υπηρεσίες.
- Ως **“υπηρεσιοστρεφής αρχιτεκτονική”** ορίζεται μια αρχιτεκτονική προσέγγιση (ή στυλ) που υποστηρίζει τον υπηρεσιοστρεφή προσανατολισμό.
- Ως **“αρχιτεκτονική προσέγγιση ή στυλ”** ορίζεται ένας συνδυασμός διακριτών χαρακτηριστικών επί των οποίων εφαρμόζεται ή εκφράζεται μια αρχιτεκτονική.

Υπηρεσίες - Ορισμός

Η υπηρεσιοστρεφής αρχιτεκτονική διαθέτει τα ακόλουθα διακριτά χαρακτηριστικά:

- Βασίζεται στον σχεδιασμό υπηρεσιών – που αντικατοπτρίζουν επιχειρησιακές δραστηριότητες του πραγματικού κόσμου – που συνθέτουν τις ενδο-οργανωσιακές ή δι-οργανωσιακές επιχειρησιακές διεργασίες.
- Κάθε αναπαράσταση υπηρεσίας αξιοποιεί περιγραφές του οργανισμού προκειμένου να παράσχει το πλαίσιο λειτουργίας (π.χ. επιχειρησιακές διεργασίες, στόχοι, κανόνες, πολιτικές, διεπαφές υπηρεσιών και συνιστώσες υπηρεσιών) και πραγματώνει υπηρεσίες χρησιμοποιώντας την έννοια της ενορχήστρωσης των υπηρεσιών.

Υπηρεσίες - Ορισμός

- Θέτει μοναδικές απαιτήσεις αναφορικά με την υποδομή – συνιστάται η χρήση ανοικτών προτύπων για την επίτευξη της διαλειτουργικότητας και της αδιαφανούς κατανομής των υπηρεσιών (ο χρήστης καλεί μια υπηρεσία χωρίς αναγκαία να γνωρίζει τον τόπο εγκατάστασής της).
  - Cloud infrastructure as the service infrastructure
    - ??? Cloud provider (Google <-> Azure)
  - Dependency on other services: SQL service kai epomenos where to deploy each service is important!
- Οι πραγματώσεις της εξαρτώνται από το περιβάλλον – περιορίζονται ή ωθούνται από το πλαίσιο λειτουργίας και πρέπει να περιγράφονται εντός αυτού του πλαισίου λειτουργίας.
- Απαιτεί ισχυρή διακυβέρνηση της αναπαράστασης και της πραγμάτωσης των υπηρεσιών.
- Απαιτεί εξονυχιστικό έλεγχο για τον καθορισμό καλών υπηρεσιών

Υπηρεσίες - Ορισμός

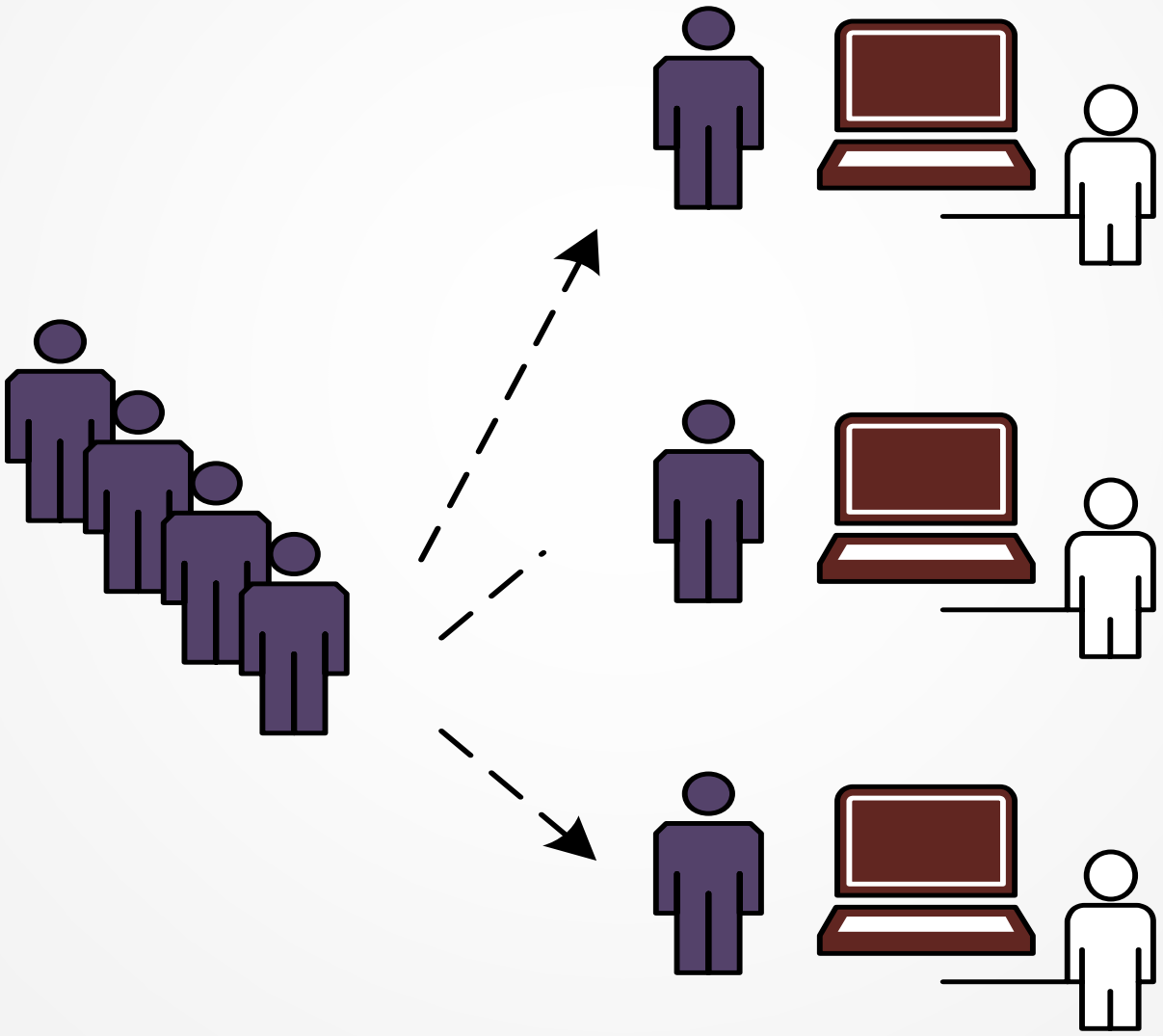
Παράδειγμα υπηρεσιοστρεφούς προσανατολισμού: Μια τράπεζα της οποίας οι υπάλληλοι παρέχουν υπηρεσίες στους πελάτες της.

- Διαφορετικοί υπάλληλοι παρέχουν διαφορετικές υπηρεσίες
- Κάποιοι υπάλληλοι είναι εκπαιδευμένοι να παρέχουν πολύ εξειδικευμένες υπηρεσίες.
- Μεταξύ των τυπικών υπηρεσιών που παρέχονται περιλαμβάνονται οι ακόλουθες:
  - Διαχείριση λογαριασμών (άνοιγμα και κλείσιμο λογαριασμών).
  - Χορηγήσεις δανείων (επεξεργασία αιτήσεων, ερωτήσεις αναφορικά με τις προϋποθέσεις, κλπ).
  - Αναλήψεις, καταθέσεις και μεταφορές.
  - Ανταλλαγή συναλλάγματος.

Υπηρεσίες - Ορισμός

- Το ίδιο σύνολο υπηρεσιών μπορεί να παρέχεται από πολλούς τραπεζικούς υπαλλήλους προκειμένου να επιτυγχάνεται, κατά το δυνατόν, ισοκατανομή του φόρτου εργασίας και υψηλή διαθεσιμότητα.
- Ο κάθε πελάτης δεν ενδιαφέρεται για το τι συμβαίνει πίσω από γκισέ αρκεί να του παρασχεθεί η επιζητούμενη υπηρεσία.
- Για την επεξεργασία μιας σύνθετης δοσοληψίας ενός πελάτη μπορεί να απαιτείται απ’ αυτόν να επισκεφθεί πολλούς υπαλλήλους εκτελώντας έτσι μια ροή επιχειρησιακής διεργασίας.

Παροχή υπηρεσιών σε πελάτες τράπεζας από τραπεζικούς υπαλλήλους

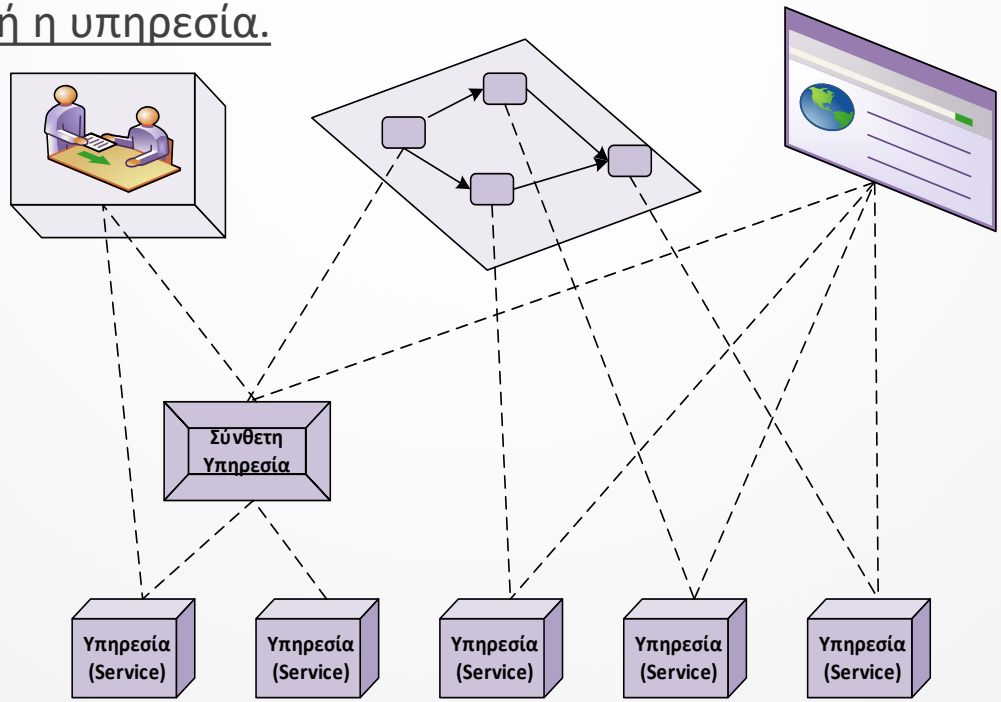


Υπηρεσίες - Ορισμός

- Πίσω από τον γκισέ λειτουργούν τα ψηφιακά συστήματα που αυτοματοποιούν τις υπηρεσίες που παρέχει η τράπεζα.
- Οι υπηρεσίες παρέχονται στους πελάτες μέσω των τραπεζικών υπαλλήλων.
- Συνεπώς:
  - Οι υπηρεσίες που έχουν πραγματωθεί μέσω ψηφιακών συστημάτων πρέπει να εναρμονίζονται και να υποστηρίζουν τις υπηρεσίες που παρέχονται από τους τραπεζικούς υπαλλήλους.
  - Μια συνεπής προσέγγιση ορισμού των ψηφιακών υπηρεσιών που εναρμονίζονται με επιχειρησιακές λειτουργίες και διεργασίες καθιστά ευκολότερη για τα ψηφιακά συστήματα να υποστηρίζουν τους επιχειρηματικούς στόχους και να προσαρμόζονται ευκολότερα στην παροχή των ίδιων υπηρεσιών μέσω ανθρώπων, ATM και του παγκόσμιου ιστού.

Υπηρεσίες - Ορισμός

- Η ίδια τραπεζική υπηρεσία μπορεί να προσπελασθεί από πελάτες της τράπεζας μέσω (taytoxrona pollapla kai cross-channel: omnichannel):
  - τραπεζικών υπαλλήλων (φυσική συναλλαγή)
  - μιας μηχανής αυτόματων συναλλαγών (ATM)
  - του παγκόσμιου ιστού (e-Banking)
- Τα περιβάλλοντα πραγμάτωσης των υπηρεσιών δεν έχουν ιδιαίτερη σημασία. Σημασία έχει αυτή καθ' αυτή η υπηρεσία.



Υπηρεσιοστρεφείς αρχιτεκτονικές

Μια υπηρεσιοστρεφής αρχιτεκτονική – ΥΣΑ *(service-oriented architecture – SOA)* είναι:

- Ένας τρόπος κατασκευής ΠΣ που εμπερικλείει μια ολοκληρωμένη άποψη των επιχειρησιακών υπηρεσιών *(business services)* ενός οργανισμού.
- Μια συλλογή από ψηφιακές υπηρεσίες ή υπηρεσίες ιστού (που αντανακλούν επιχειρησιακές υπηρεσίες) οι οποίες επικοινωνούν μεταξύ τους με κάποιον τρόπο ανεξάρτητο τεχνολογίας και κατασκευαστού.

Πλεονεκτήματα και οφέλη της ΥΣΑ

- **Φορητότητα του κώδικα:** Πρόκειται για ακόμη ένα σημαντικό όφελος της ΥΣΑ και καθίσταται δυνατό διότι ο τόπος εγκατάστασης του λογισμικού είναι αδιαφανής σε μια ΥΣΑ. Οι πιο πολλοί πελάτες δεν νοιάζονται που έχουν εγκατασταθεί οι υπηρεσίες λόγω της παροχής δυναμικής δίπλωσης και αναζήτησης των υπηρεσιών. Αυτό σημαίνει ότι παρέχεται η δυνατότητα στους οργανισμούς που χρησιμοποιούν ΥΣΑ να μετακινούν υπηρεσίες σε διαφορετικές μηχανές ή να καταφεύγουν σε εξωτερικούς παρόχους υπηρεσιών.
- **Λιγότερα ελαττώματα:** Η πιθανότητα εμφάνισης ελαττωμάτων είναι πολύ μικρότερη καθότι καθίσταται κατά πολύ ευκολότερη η διενέργεια πολλαπλών δοκιμών λόγω, κυρίως, των ευκολότερων δοκιμών των δημοσιευμένων διεπαφών των υπηρεσιών. Η δυνατότητα διενέργειας πολλαπλών δοκιμών συνεπάγεται αφενός την επίτευξη υψηλότερου επιπέδου ακρίβειας των υπηρεσιών και αφετέρου πολύ λιγότερα σφάλματα.

Πλεονεκτήματα και οφέλη της ΥΣΑ

- **Η επανάχρηση:** Παρέχεται η δυνατότητα εύκολης επανάχρησης υπηρεσιών κατά τη λειτουργία και η ευκολία είναι τόσο μεγάλη όσο η εύρεση μιας υπηρεσίας σε έναν κατάλογο και δίπλωσής της σ' αυτόν. Οι κατασκευαστές δεν χρειάζεται να ανησυχούν για τις πλατφόρμες ή άλλες ασυμβατότητες.
- **Υποστήριξη διαφόρων ειδών πελατών:** Κάθε οργανισμός μπορεί να εξουσιοδοτήσει πολλά είδη πελατών για πρόσβαση σε μια υπηρεσία. Υα επίπεδα έχουν διαιρεθεί σε επίπεδα υπηρεσιών και πελατών οπότε είναι ευκολότερη η πραγμάτωσή τους.
- **Υψηλότερο επίπεδο διαθεσιμότητας:** Χρήση πολλών εξυπηρετητών λόγω του γεγονότος ότι οι ΥΣΑ υποστηρίζουν την αδιαφανή τοποθεσία τους. Για παράδειγμα, αν διακοπεί η λειτουργία ή παρουσιαστεί κάποιο πρόβλημα σε έναν υπολογιστή ή σε ένα τμήμα του δικτύου, τα αιτήματα πρόσβασης μπορούν να οδηγηθούν σε άλλες υπηρεσίες χωρίς να το γνωρίζει ο πελάτης ή να απασχολείται με αυτό το θέμα.

Προκλήσεις για την υιοθέτηση των ΥΣΑ

- Πρόκληση είναι και το γεγονός ότι μπορεί να απαιτείται η τροποποίηση μερικών υπαρχόντων εφαρμογών προκειμένου να καταστεί δυνατή η συμμετοχή τους σε μια ΥΣΑ.
  - Μερικές εφαρμογές μπορεί να μην διαθέτουν επικλήσιμες διεπαφές που να είναι δυνατό να χρησιμοποιηθούν ως υπηρεσίες.
    - SOAP (problematic case)
    - REST – JSON
    - GraphQL: queries to interfaces
  - Άλλες εφαρμογές είναι προσβάσιμες μόνο μέσω μεταφοράς αρχείου ή μέσω εισροών/εκροών δεδομένων σε μορφή batch και μπορεί να χρειάζονται πρόσθετες εφαρμογές για να καταστεί δυνατή η συμμετοχή τους σε μια ΥΣΑ.

Interface

- Synolo methodon gia to interaction me to service
- P.x. service **weatherPrediction**
  - provideTemperature(City c)
  - getPrediction(City c, Window tw)
    - REST interface: Athens, Mon mor 8 C, mon mid 12 C, mon aft 9, mon aver 9
    - GraphQL: Athens, mon aver 9

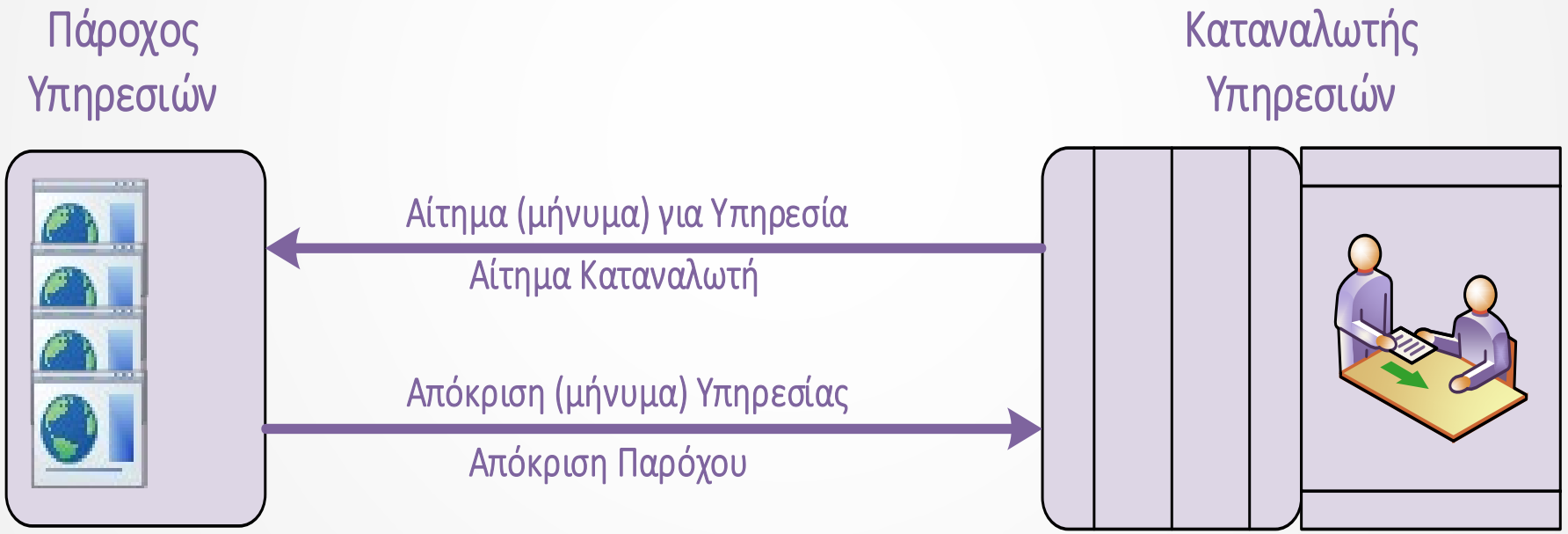
Περιεχόμενα

- Σύνοψη προηγούμενης διάλεξης
- Λογική Ενοποιημένη Διεργασία
- Ανάπτυξη συστημάτων συνιστωσών λογισμικού
  - Version Control
  - Σύνθεση των ΠΣ που βασίζονται σε συνιστώσες
  - Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών
- Υπηρεσιοστρεφής ανάπτυξη συστημάτων
  - Η χρήση των υπηρεσιών ιστού
- DevOps

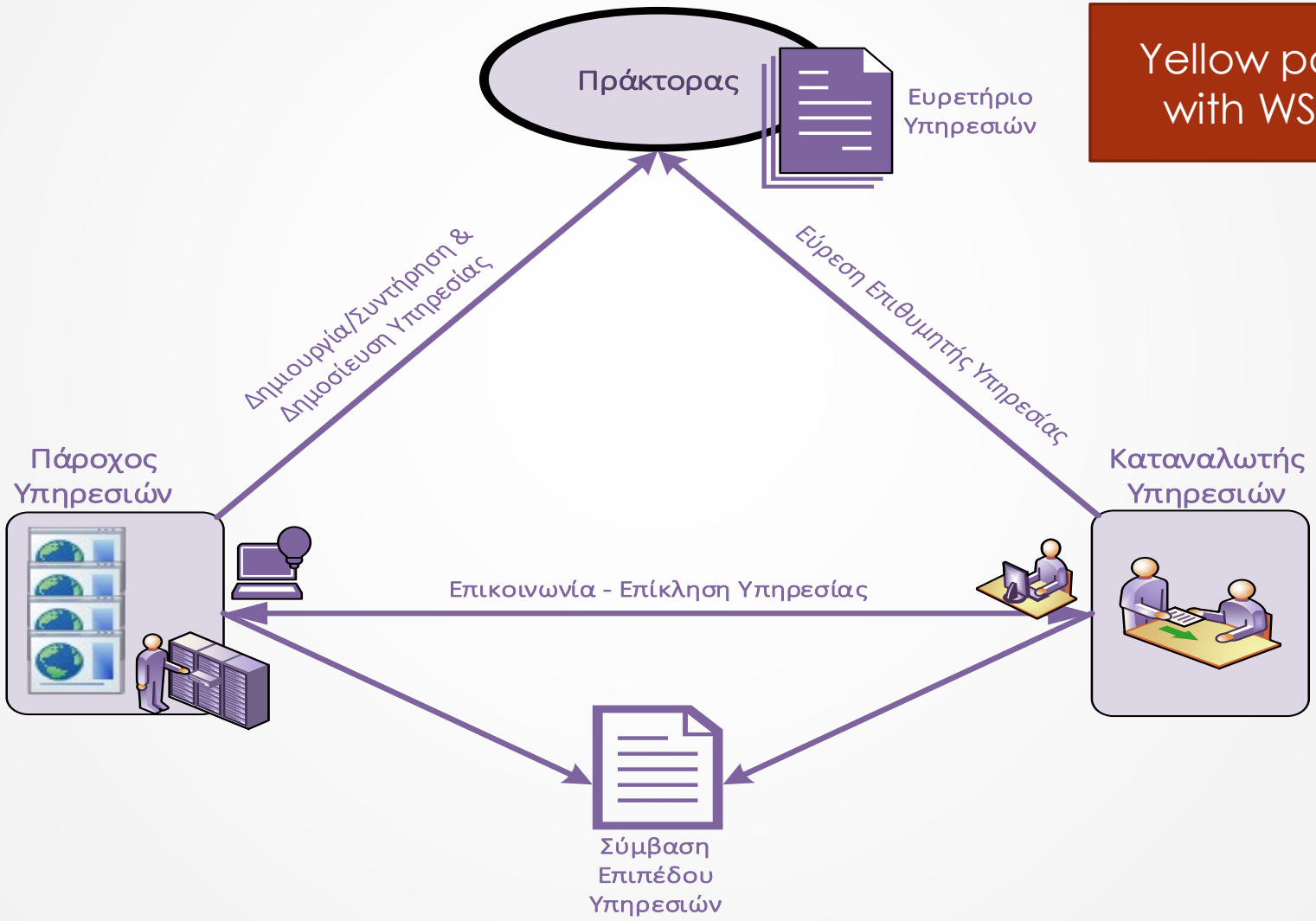
Η χρήση των υπηρεσιών ιστού

- Οι υπηρεσίες ιστού είναι η πρόταση μιας διαφορετικής προσέγγισης για την επίλυση προβλημάτων, ειδικά προβλημάτων ολοκλήρωσης εφαρμογών/συστημάτων, λόγω των νέων δυνατοτήτων που παρέχει η τεχνολογία αυτή.
- Οι υπηρεσίες ιστού εκτελούνται μέσω ανταλλαγής μηνυμάτων σύμφωνα με ένα ή περισσότερα πρότυπα ανταλλαγής μηνυμάτων *(message exchange patterns - MEPs)*, όπως:
  - Αίτημα/απόκριση *(request/response)*
  - Μονόδρομο ασύγχρονο *(one-way asynchronous)*
  - Δημοσίευση/συνδρομή *(publish/subscribe)*

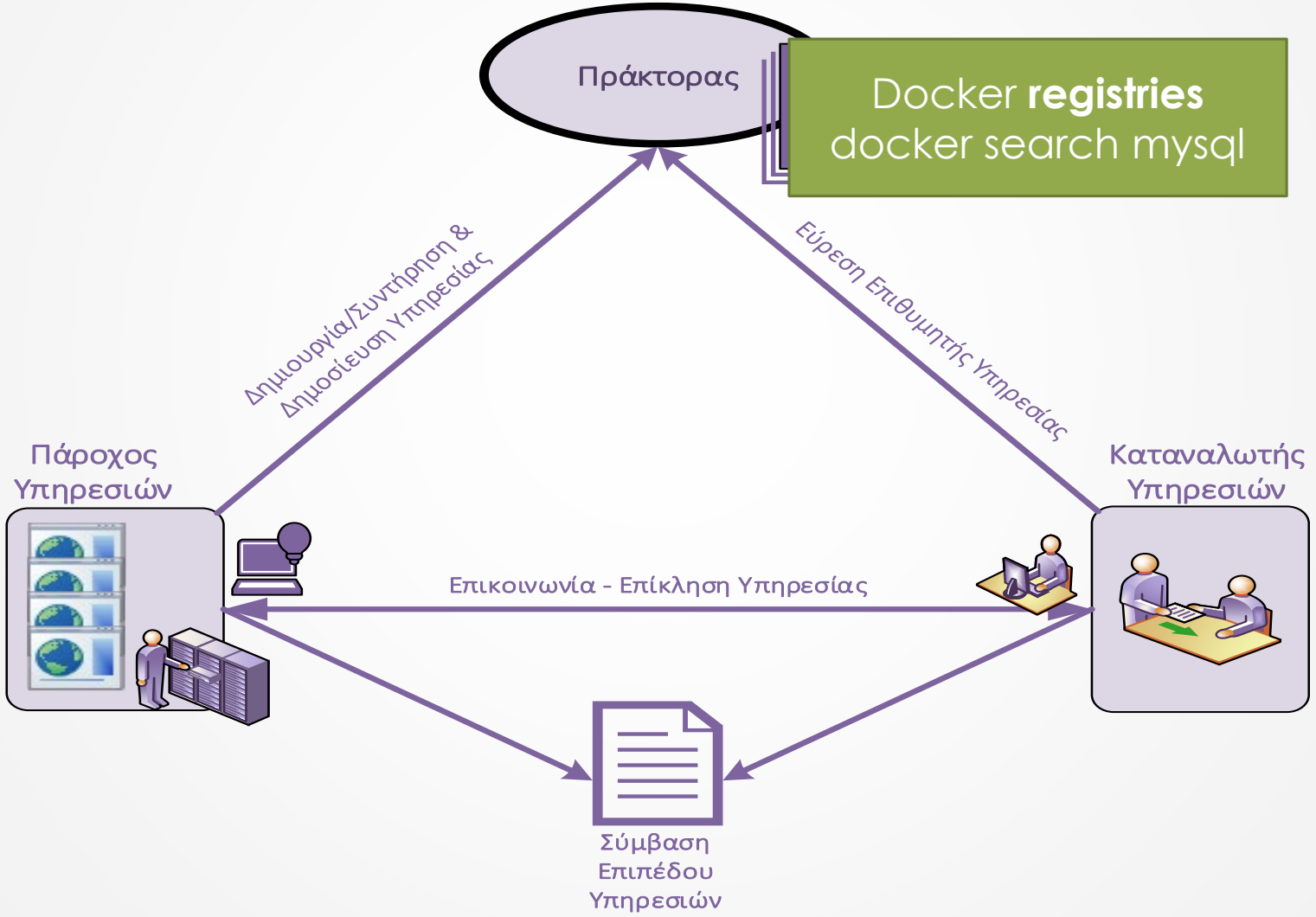
Διασύνδεση καταναλωτή – παρόχου υπηρεσιών χρησιμοποιώντας τεχνολογίες που είναι κατανοητές και στους δύο



Επίκληση υπηρεσιών εντός μιας ΥΣΑ: ..2010



Επίκληση υπηρεσιών εντός μιας ΥΣΑ: 2018+



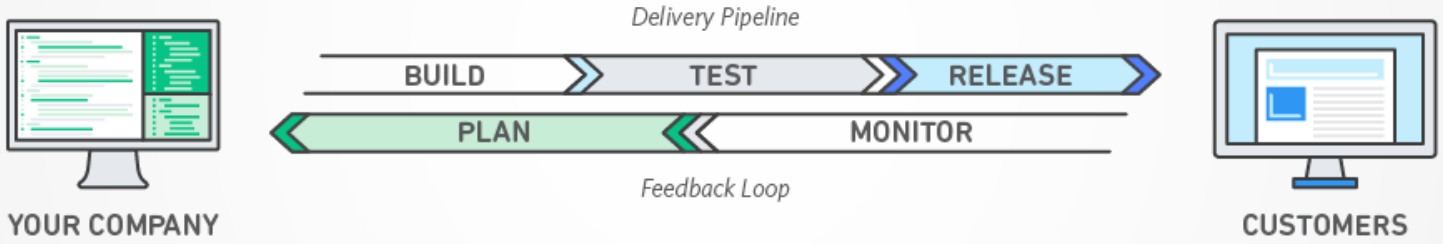
Περιεχόμενα

- Σύνοψη προηγούμενης διάλεξης
- Λογική Ενοποιημένη Διεργασία
- Ανάπτυξη συστημάτων συνιστωσών λογισμικού
  - Version Control
  - Σύνθεση των ΠΣ που βασίζονται σε συνιστώσες
  - Κύκλος ζωής ανάπτυξης συστημάτων συνιστωσών
- Υπηρεσιοστρεφής ανάπτυξη συστημάτων
  - Η χρήση των υπηρεσιών ιστού
- DevOps

DevOps

- Το DevOps είναι ο συνδυασμός πολιτιστικών φιλοσοφιών, πρακτικών και εργαλείων που αυξάνουν την **ικανότητα μιας επιχείρησης να παρέχει εφαρμογές και υπηρεσίες σε υψηλή ταχύτητα:** εξελίσσονται και βελτιώνονται τα προϊόντα με ταχύτερο ρυθμό από ό,τι οι οργανισμοί που χρησιμοποιούν παραδοσιακές διαδικασίες ανάπτυξης λογισμικού και διαχείρισης υποδομών.
- Η ταχύτητα αυτή επιτρέπει στους οργανισμούς να εξυπηρετούν καλύτερα τους πελάτες τους και να ανταγωνίζονται πιο αποτελεσματικά στην αγορά.

DevOps


Delivery Pipeline
BUILD
TEST
RELEASE
PLAN
MONITOR
Feedback Loop
YOUR COMPANY
CUSTOMERS

Πως λειτουργεί το μοντέλο

- Σύμφωνα με ένα μοντέλο DevOps, οι ομάδες ανάπτυξης και λειτουργίας δεν είναι πλέον απομονωμένες
  - Συνήθως οι δύο αυτές ομάδες συγχωνεύονται σε μία ομάδα όπου οι μηχανικοί εργάζονται σε ολόκληρο τον κύκλο ζωής της εφαρμογής, από την ανάπτυξη και δοκιμή μέχρι την ανάπτυξη σε λειτουργίες, δεξιοτήτων που δεν περιορίζονται σε μία μόνο λειτουργία
- Αυτές οι ομάδες χρησιμοποιούν πρακτικές για την αυτοματοποίηση διαδικασιών
  - **Developer: code, push git, docker build**
    - Tester: integrate (Jenkins), testing (junit), bug reporting (gitlab)
  - **Operations: get git, docker build, docker push (production environment)**
- Χρησιμοποιούν μια στοίβα τεχνολογίας και εργαλεία που τους βοηθούν να λειτουργούν και να εξελίσσονται εφαρμογές γρήγορα και αξιόπιστα
- Όταν η ασφάλεια είναι η εστίαση όλων σε μια ομάδα DevOps, αυτό μερικές φορές αναφέρεται ως DevSecOps

Πλεονεκτήματα

- Ταχύτητα
- Γρήγορη Παράδοση
- Αξιοπιστία
- Κλιμάκωση
- Αποτελεσματικότητα
- Ασφάλεια

Ταχύτητα

- Δίνει τη δυνατότητα η ομάδα να κινείται με μεγάλη ταχύτητα, ώστε να μπορεί να ενσωματώνει γρήγορα καινοτομίες και να προσαρμόζεται καλύτερα στις αλλαγές
- Το μοντέλο DevOps επιτρέπει στους προγραμματιστές και τις ομάδες λειτουργίας επιτύχουν πιο εύκολα αυτά τα αποτελέσματα. Για παράδειγμα, η χρήση microservices και CI/CD επιτρέπουν στις ομάδες να υλοποιούν γρήγορα τις υπηρεσίες και στη συνέχεια να τις διαθέτουν άμεσα στους τελικούς χρήστες
- Παράλληλα παρακολουθούν τις εκδόσεις του ΠΣ που έχει διατεθεί στους χρήστες, αξιολογεί τα σχόλιά τους και ενσωματώνει τις σχετικές ενημερώσεις γρηγορότερα

Αξιοπιστία

- Μέσα από τη χρήση πρακτικών όπως η συνεχής ολοκλήρωση και η συνεχής παράδοση, μπορούμε με αυτόματο τρόπο να ελέγξουμε ότι κάθε αλλαγή είναι λειτουργική και ασφαλής
- Οι πρακτικές παρακολούθησης και καταγραφής βοηθούν επίσης να έχουμε μια πλήρη εικόνα για την απόδοση της εφαρμογής σε πραγματικό χρόνο
- Με αυτό τον τρόπο εξασφαλίζουμε την ποιότητα των ενημερώσεων εφαρμογών και των αλλαγών της υποδομής, ώστε να προσφέρουμε αξιόπιστα, με ταχύτερο ρυθμό τις υπηρεσίες, διατηρώντας ταυτόχρονα μια θετική εμπειρία για τους τελικούς χρήστες

Γρήγορη Παράδοση

- Με την αύξηση του ρυθμού νέων εκδόσεων του ΠΣ και των εφαρμογών δίνεται η δυνατότητα για καινοτόμες λειτουργίες και πολλές βελτιώσεις του προϊόντος ή των υπηρεσιών.
- Όσο πιο γρήγορα προστεθούν οι νέες δυνατότητες και διορθωθούν τα σφάλματα, τόσο πιο γρήγορα μπορούμε να ανταποκριθούμε στις ανάγκες των πελατών και να δημιουργήσουμε ανταγωνιστικό πλεονέκτημα.

Κλιμάκωση

- Η αυτοματοποίηση βοηθά να διαχειριστούμε πολύπλοκα ή μεταβαλλόμενα συστήματα αποτελεσματικά και με μειωμένο κίνδυνο
- Η χρήση νέων τεχνολογιών βοηθά να διαχειριστούμε τα περιβάλλοντα ανάπτυξης, δοκιμών και παραγωγής με έναν επαναλαμβανόμενο και αποδοτικότερο τρόπο
- Ανάπτυξη και εγκατάσταση υπηρεσιών στην υποδομή με κλιμάκωση των απαιτούμενων πόρων (γνώση υπηρεσιών / εφαρμογών)

Αποτελεσματικότητα

- Με το μοντέλο DevOps μπορούμε να δημιουργήσουμε πιο αποτελεσματικές ομάδες το οποίο δίνει έμφαση σε αξίες όπως η ιδιοκτησία και η λογοδοσία
- Οι ομάδες ανάπτυξης και λειτουργίας συνεργάζονται στενά, μοιράζονται πολλές ευθύνες και συνδυάζουν τις ροές εργασίας τους
- Αυτό μειώνει τις ανεπάρκειες και εξοικονομεί χρόνο (π.χ. μειωμένες περιόδους παράδοσης μεταξύ προγραμματιστών και λειτουργιών, γραφή κώδικα που λαμβάνει υπόψη το περιβάλλον στο οποίο εκτελείται)

Η αξία των DevOps

- Το λογισμικό και το Διαδίκτυο έχουν μεταμορφώσει τον κόσμο και τις βιομηχανίες
- Το λογισμικό δεν υποστηρίζει απλώς μια επιχείρηση, αλλά γίνεται αναπόσπαστο συστατικό μέρος κάθε τμήματος μιας επιχείρησης
- Οι εταιρείες αλληλεπιδρούν με τους πελάτες τους μέσω λογισμικού που παρέχεται ως ηλεκτρονικές υπηρεσίες ή εφαρμογές και σε όλα τα είδη συσκευών
- Η μεθοδολογία DevOps αποσκοπεί στην απομάκρυνση των φραγμών μεταξύ δύο παραδοσιακά ξεχωριστών ομάδων, ανάπτυξης και λειτουργίας
  - Οι δύο ομάδες συνεργάζονται για να βελτιστοποιήσουν τόσο την παραγωγικότητα των προγραμματιστών όσο και την αξιοπιστία των λειτουργιών
  - Προσπαθούν να επικοινωνούν συχνά, να αυξάνουν την αποδοτικότητα και να βελτιώνουν την ποιότητα των υπηρεσιών που παρέχουν στους πελάτες.
- Οι οργανισμοί που χρησιμοποιούν ένα μοντέλο DevOps, ανεξάρτητα από την οργανωτική δομή τους, έχουν ομάδες που βλέπουν ολόκληρο τον κύκλο ανάπτυξης και του κύκλου ζωής της υποδομής ως μέρος των ευθυνών τους

Πρακτικές DevOps

- Continuous Integration
- Continuous Delivery
- Microservices
- Infrastructure as Code
- Monitoring and Logging
- Communication and Collaboration

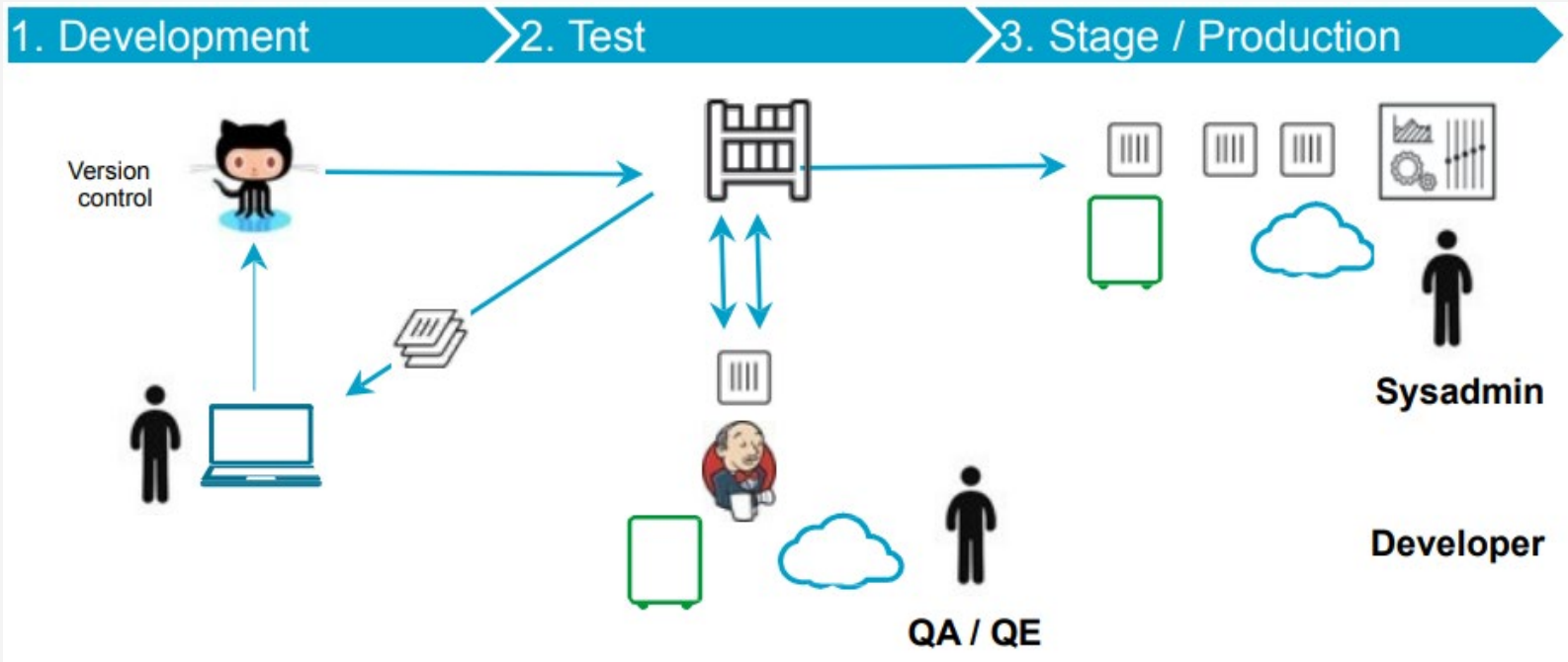
Συνεχής ολοκλήρωση

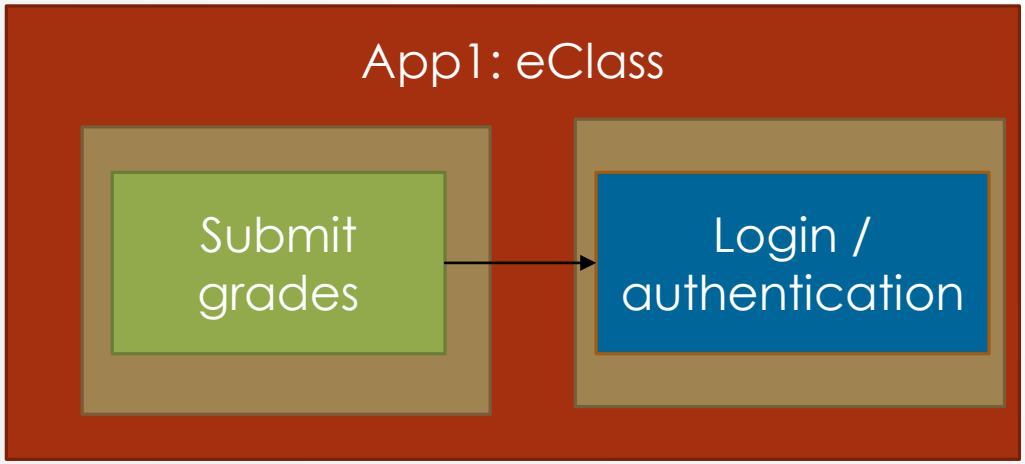
- Η συνεχής ολοκλήρωση (Continuous Integration) είναι μια πρακτική ανάπτυξης λογισμικού, όπου οι προγραμματιστές συγχωνεύουν τακτικά τις αλλαγές κώδικα τους σε ένα κεντρικό αποθετήριο, μετά τον οποίο εκτελούνται αυτοματοποιημένες κατασκευές και δοκιμές
- Οι βασικοί στόχοι της συνεχιζόμενης ολοκλήρωσης είναι η ταχύτερη εύρεση και αντιμετώπιση σφαλμάτων, η βελτίωση της ποιότητας του λογισμικού και η μείωση του χρόνου που απαιτείται για την επικύρωση και την απελευθέρωση νέων ενημερώσεων λογισμικού

Συνεχής παράδοση

- Η συνεχής παράδοση (continuous delivery) είναι μια πρακτική ανάπτυξης λογισμικού όπου οι αλλαγές κώδικα δημιουργούνται αυτόματα, δοκιμάζονται και προετοιμάζονται για την παραγωγή
- Διευρύνει τη συνεχή ενσωμάτωση αναπτύσσοντας όλες τις αλλαγές κώδικα σε ένα περιβάλλον δοκιμών ή / και ένα περιβάλλον παραγωγής μετά το στάδιο κατασκευής
- Όταν η συνεχής παράδοση υλοποιείται σωστά, οι προγραμματιστές θα έχουν πάντοτε ένα έτοιμο προς ανάπτυξη έμβλημα που έχει περάσει από μια τυποποιημένη διαδικασία δοκιμής

Continuous Integration and Delivery





Microservices

- Η αρχιτεκτονική microservices είναι μια προσέγγιση σχεδιασμού για την κατασκευή μιας ενιαίας εφαρμογής ως ενός συνόλου μικρών υπηρεσιών
- Κάθε υπηρεσία εκτελείται με τη δική της διαδικασία και επικοινωνεί με άλλες υπηρεσίες μέσω ενός σαφώς καθορισμένου περιβάλλοντος εργασίας χρησιμοποιώντας ένα ελαφρύ μηχανισμό, συνήθως μια διεπαφή προγραμματισμού εφαρμογών (API) που βασίζεται σε HTTP
- Μπορούμε να χρησιμοποιήσουμε διαφορετικά πλαίσια ή γλώσσες προγραμματισμού για να υλοποιήσουμε μικροεπιχειρήσεις και να τις αναπτύξουμε ανεξάρτητα, ως ενιαία υπηρεσία ή ως ομάδα υπηρεσιών

Υποδομή ως Κώδικας

- Η υποδομή ως κώδικας (Infrastructure as Code) είναι μια πρακτική στην οποία η υποδομή παρέχεται και διαχειρίζεται χρησιμοποιώντας τεχνικές ανάπτυξης κώδικα και λογισμικού, όπως ο έλεγχος εκδόσεων και η συνεχής ενσωμάτωση
  - Cloud infrastructure = VMs / Containers
  - Call method of infrastructure provider and get the infrastructure (e.g. newVM())
- Το μοντέλο που βασίζεται στο API του νέφους επιτρέπει στους προγραμματιστές και τους διαχειριστές συστημάτων να αλληλεπιδράσουν προγραμματικά με την υποδομή και σε κλίμακα, αντί να χρειάζεται να ρυθμίζουν με μη αυτόματο τρόπο τους πόρους
- Έτσι, οι μηχανικοί μπορούν να διασυνδέονται με την υποδομή χρησιμοποιώντας εργαλεία που βασίζονται σε κώδικα και αντιμετωπίζουν την υποδομή με τρόπο παρόμοιο με τον τρόπο με τον οποίο αντιμετωπίζουν τον κώδικα εφαρμογής
- Επειδή ορίζονται από τον κώδικα, η υποδομή και οι διακομιστές μπορούν γρήγορα να αναπτυχθούν χρησιμοποιώντας τυποποιημένα μοτίβα, ενημερωμένα με τα πιο πρόσφατα μπαλώματα και εκδόσεις ή αντιγραφή σε επαναλαμβανόμενους τρόπους

Παρακολούθηση και Καταγραφή

- Οι οργανισμοί παρακολουθούν τις μετρήσεις και τα αρχεία καταγραφής για να δουν πώς επηρεάζουν την απόδοση των εφαρμογών και των υποδομών την εμπειρία του τελικού χρήστη του προϊόντος τους
- Καταγράφοντας, κατηγοριοποιώντας και αναλύοντας τα δεδομένα και τα αρχεία καταγραφής που δημιουργούνται από εφαρμογές και υποδομές, οι οργανισμοί κατανοούν πώς οι αλλαγές ή οι ενημερώσεις επηρεάζουν τους χρήστες, εξαλείφοντας τις βαθύτερες αιτίες των προβλημάτων ή τις απρόσμενες αλλαγές
- Η ενεργή παρακολούθηση καθίσταται ολοένα και πιο σημαντική καθώς οι υπηρεσίες πρέπει να είναι διαθέσιμες 24 ώρες το 24ωρο και 7 ημέρες την εβδομάδα καθώς αυξάνεται η συχνότητα επικαιροποίησης εφαρμογών και υποδομών η δημιουργία ειδοποιήσεων ή η ανάλυση σε πραγματικό χρόνο αυτών των δεδομένων βοηθά τους οργανισμούς να παρακολουθούν πιο προληπτικά τις υπηρεσίες τους

Επικοινωνία και Συνεργασία

- Η αυξημένη επικοινωνία και συνεργασία σε έναν οργανισμό είναι μία από τις βασικές πτυχές των DevOps
- Η χρήση DevOps εργαλείων και αυτοματοποίησης της διαδικασίας παράδοσης λογισμικού καθιερώνει τη συνεργασία, συνδυάζοντας φυσικά τις ροές εργασίας και τις ευθύνες ανάπτυξης και λειτουργίας
- Επιπλέον, οι ομάδες αυτές θέτουν ισχυρούς πολιτιστικούς κανόνες σχετικά με την ανταλλαγή πληροφοριών και τη διευκόλυνση της επικοινωνίας μέσω της χρήσης εφαρμογών συνομιλίας, συστημάτων παρακολούθησης έργων ή έργων και wikis
- Αυτό βοηθά στην επιτάχυνση της επικοινωνίας μεταξύ των προγραμματιστών, των επιχειρήσεων και ακόμη και άλλων ομάδων όπως το μάρκετινγκ ή τις πωλήσεις, επιτρέποντας σε όλα τα μέρη της οργάνωσης να ευθυγραμμιστούν πιο στενά με τους στόχους και τα έργα