



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Ηλεκτρολόγων Μηχανικών  
και Μηχανικών Υπολογιστών

Μονάδα Επεξεργασίας Σήματος  
και Βιοϊατρικής Τεχνολογίας

## Διπλωματική Εργασία

---

# Υλοποίηση διεπαφής έξυπνου καθρέφτη για ανάπτυξη εφαρμογών lifestyle και εξατομικευμένης υγείας

---

Ζαχαρόπουλος Φίλιππος 8559  
filipposz@ece.auth.gr

Παπαγεωργίου Δημήτριος 8884  
dim\_papag@windowslive.com

*Επιβλέποντες:*

Χατζηλεοντιάδης Λεόντιος  
Καθηγητής Α.Π.Θ.

Ντράχα Αναστασία  
Μεταδιδακτορική Ερευνήτρια

Χατζηδημητρίου Στέλιος  
Μεταδιδακτορικός Ερευνητής

13 Ιανουαρίου 2022



---

## ΕΥΧΑΡΙΣΤΙΕΣ

---

Πρώτο από όλους θα ήθελα να ευχαριστήσω τον Κωνσταντίνο Παναγιώτου, ο οποίος με βοήθησε καθόλη τη διάρκεια της διπλωματικής, με κατεύθυνε όπως έπρεπε και ήταν πάντα διαθέσιμος για να με βοηθάει σε προβλήματα που αντιμετώπιζα. Επίσης θα ήθελα να ευχαριστήσω τον κ. Ανδρέα Συμεωνίδη που μου έδωσε τη δυνατότητα να αναλάβω αυτή τη διπλωματική και να ασχοληθώ με το αντικείμενο που πάντα με ενδιέφερε, αλλά και τους Μάνο Τσαρδούλια και Αλέξανδρο Φιλοθέου που μου παρείχαν τις γνώσεις τους στον τομέα της ρομποτικής, και μπόρεσα μέσω της ομάδας R4A να γνωρίσω και να ασχοληθώ με ένα εξίσου ενδιαφέρον αντικείμενο.

Πέραν των επιστημονικών συνεργατών, θα ήθελα να ευχαριστήσω τους φίλους και τις φίλες μου για τη συμπαράστασή τους και για όλες τις όμορφες στιγμές που ζήσαμε στην περίοδο των φοιτητικών μας χρόνων, αλλά και τους γονείς μου, Βαγγέλη και Ζωή, και την αδερφή μου, Ευαγγελία, για την στήριξη και τα εφόδια που μου έδωσαν και που ήταν στο πλευρό μου όποτε τους χρειαζόμουν.



---

## Περίληψη

Το διαδίκτυο των πραγμάτων (*Internet of Things* ή *IoT*) είναι ένας κλάδος που εξελίσσεται ραγδαία ειδικά τα τελευταία χρόνια. Υπάρχει η δυνατότητα ανάπτυξης όλο και περισσότερων εφαρμογών, χρήσιμες για πολλούς ανθρώπους, είτε έχουν να κάνουν με απλές λειτουργίες σε συστήματα αυτοματισμού, είτε με μεγαλύτερης κλίμακας εφαρμογές στη βιομηχανία. Επομένως, όλο και περισσότερος κόσμος επιθυμεί να ασχοληθεί με το αντικείμενο αυτό.

Η διαδικασία υλοποίησης ενός IoT συστήματος περιλαμβάνει την ανάπτυξη κώδικα για τον έλεγχο των συσκευών. Μάλιστα, στις περισσότερες περιπτώσεις η γρήγορη απόκριση είναι υψίστης σημασίας, επομένως απαιτείται η ανάπτυξη χαμηλού επιπέδου κώδικα, καθώς και η χρήση λειτουργικών συστημάτων πραγματικού χρόνου (*Real Time Operating System* ή *RTOS*). Επίσης, λόγω της μεγάλης ετερογένειας IoT συσκευών που υπάρχουν στην αγορά, κρίνεται αναγκαία η κατανόηση των δυνατοτήτων που η εκάστοτε συσκευή μπορεί να προσφέρει, ώστε να γίνεται η κατάλληλη επιλογή τους, προσαρμοσμένη στις ανάγκες του συστήματος προς υλοποίηση.

Οι ενέργειες αυτές είναι λογικό να φαίνονται περίπλοκες σε κάποιους χρήστες, ειδικότερα στα άτομα που είναι τεχνολογικά ακατάρτιστα, δεν έχουν δηλαδή τις απαραίτητες προγραμματιστικές γνώσεις, αλλά παρόλα αυτά επιθυμούν να κατασκευάσουν ένα IoT σύστημα π.χ. για προσωπική τους χρήση. Αυτό έχει ως αποτέλεσμα μεγάλη μερίδα κόσμου που θέλει να ασχοληθεί με το IoT να αποθαρρύνεται.

Η μοντελοστρεφής μηχανική (*Model Driven Engineering* ή *MDE*), έρχεται να δώσει λύση στα προβλήματα που μπορεί να αντιμετωπίσουν όσοι/ες θέλουν να ασχοληθούν με το IoT, αλλά και γενικότερα να απλοποιήσει τη διαδικασία παραγωγής λογισμικού, καθώς μπορεί να παρέχει την ανάπτυξη IoT συστημάτων σε ένα πιο αφαιρετικό επίπεδο, το οποίο είναι πιο φιλικό προς τον απλό χρήστη.

Μέσω της παρούσας διπλωματικής εργασίας, δίνεται η δυνατότητα σε κάποιον/α να περιγράψει, με χρήση μοντέλων, IoT συσκευές, μέσω δύο γλωσσών συγκεκριμένου πεδίου (*Domain Specific Language* ή *DSL*) που αναπτύχθηκαν, για την περιγραφή των συσκευών και των μεταξύ τους συνδέσεων. Από τα μοντέλα πραγματοποιείται ένας Model-to-Text μετασχηματισμός για την αυτόματη παραγωγή λογισμικού, για μια πληθώρα IoT συσκευών, προσαρμοσμένη στα χαρακτηριστικά που επιθυμεί ο χρήστης να έχει. Το λογισμικό ελέγχου των IoT συσκευών που παράγεται υλοποιεί την λήψη μετρήσεων από αισθητήρες και την αποστολή τους σε κάποιον μεσολαβητή (*broker*), αλλά και τον έλεγχο ενεργοποιητών μέσω του *broker*. Επίσης αποτελείται από χαμηλού επιπέδου κώδικα, καθώς έχει σχεδιαστεί σύμφωνα με τις απαιτήσεις ενός λειτουργικού συστήματος πραγματικού χρόνου, το RIOT. Τέλος, πραγματοποιείται και ένας Model-to-Model μετασχηματισμός για την παραγωγή διαγραμμάτων τα οποία βοηθούν στην οπτικοποίηση και άρα καλύτερη αντίληψη από τον χρήστη για τη συνδεσμολογία και ενδοεπικοινωνία του συστήματός του.



---

# Title

## Model-driven development for low-consumption real-time IOT devices

### Abstract

Athanasios Manolis  
Intelligent Systems and Software Engineering Labgroup (ISSEL)  
Electrical & Computer Engineering Department,  
Aristotle University of Thessaloniki, Greece  
September 2021

Internet of Things (*IoT*) is a field that is evolving rapidly, especially in recent years. There is the possibility of developing even more applications which prove to be useful for many people, whether they have to do with simple functions in automation systems, or with larger scale applications in the industry. Therefore, more and more people want to work in this field.

The process of developing an IoT system involves code development to control the system's devices. In fact, in most cases fast response is of the utmost importance, so low-level code development is required, as well as the use of real-time operating systems (*RTOS*). Also, due to the great heterogeneity of IoT devices on the market, it is necessary to understand the capabilities that each device can offer, in order to make the appropriate choice of one, tailored to the needs of the system to be implemented.

These requirements may seem complicated to some users, especially to people who are technologically untrained, i.e. do not have the necessary programming skills, but still want to build an IoT system e.g. for their personal use. This results in a large portion of people wanting to get involved with IoT, being discouraged to do so.

Model Driven Engineering (*MDE*) is here to solve the problems that, those who want to get involved with IoT, may face, but also to simplify the software production process in general, as it can provide the developing of IoT systems to a more abstract level, which is more user friendly.

Through this diploma thesis, one is given the opportunity to describe, using models, IoT devices, through two domain specific Languages (*DSL*) developed for the description of devices and the connections between them. From the models, a Model-to-Text (*M2T*) transformation is performed for the automated code generation, for a variety of IoT devices, adapted to the characteristics that the user wishes for it to have. The software for controlling the IoT devices that is produced implements the process of taking measurements from sensors, and sending them to a *broker*, but also the process of controlling actuators through the broker. It also consists of low-level code, as it has been designed according to the requirements of a real time operating system, named RIOT. Finally, a Model-to-Model (*M2M*) transformation takes place in order

---

to produce diagrams that provide a visualization and thus a better understanding by the user, of the wiring and intercommunication of their system.





# Περιεχόμενα

Ευχαριστίες . . . . .	i
Περίληψη . . . . .	iii
Abstract . . . . .	v
Ακρωνύμια . . . . .	xi
<b>1 Εισαγωγή . . . . .</b>	<b>1</b>
1.1 Περιγραφή του Προβλήματος . . . . .	2
1.2 Σκοπός - Συνεισφορά της Διπλωματικής Εργασίας . . . . .	2
1.3 Διάρθρωση της Αναφοράς . . . . .	3
<b>2 Θεωρητικό Υπόβαθρο - Αρχιτεκτονική . . . . .</b>	<b>4</b>
2.1 Λειτουργικά Συστήματα . . . . .	4
2.1.1 Το ΛΣ ως Διεπαφή Χρήστη/Υπολογιστή . . . . .	4
2.1.2 Το ΛΣ ως Διαχειριστής Πόρων . . . . .	7
2.2 Open Graphics Library . . . . .	7
2.2.1 OpenGL . . . . .	7
2.3 Διαδίκτυο των Πραγμάτων . . . . .	8
2.3.1 Πώς δουλεύει το IoT . . . . .	9
2.4 Ανάπτυξη Λογισμικού . . . . .	11
2.4.1 Τεχνολογία Λογισμικού . . . . .	13
2.4.2 Ροές διαδικασίας Τεχνολογίας Λογισμικού . . . . .	14
2.4.3 Μοντέλα ανάπτυξης λογισμικού . . . . .	16
2.4.4 Σχεδιαστικά Πρότυπα . . . . .	19
2.5 Εκτίμηση Πόζας . . . . .	19
2.5.1 Μεθοδολογίες εκτίμησης πόζας . . . . .	20
2.5.2 Μοντελοποίηση ανθρώπινου σώματος . . . . .	21
2.5.3 Κατηγοριοποιήσεις λύσεων του προβλήματος εκτίμησης πόζας . . . . .	23
<b>Βιβλιογραφία . . . . .</b>	<b>30</b>

# Κατάλογος Σχημάτων

2.1	Δομή Υλικού και Λογισμικού Υπολογιστή . . . . .	5
2.2	Διαδικασία απεικόνισης γραφικών της OpenGL . . . . .	8
2.3	Αρχιτεκτονική 3 Επιπέδων . . . . .	10
2.4	Αρχιτεκτονική 5 Επιπέδων . . . . .	11
2.5	Σχετική κατανομή κόστους υλικού/λογισμικού (Πηγή: [1]) . . . . .	12
2.6	Καμπύλη αποτυχίας για το λογισμικό . . . . .	13
2.7	Ροές διαδικασίας λογισμικού . . . . .	15
2.8	Αυξητικό Μοντέλο . . . . .	17
2.9	Σπειροειδές Μοντέλο . . . . .	18
2.10	Παράδειγμα εκτίμησης πόζας . . . . .	20
2.11	Μοντέλο Σκελετού . . . . .	22
2.12	Μοντελοποίηση SMPL . . . . .	23
2.13	Μοντελοποίηση DensePose . . . . .	23
2.14	Άρθρα εκτίμησης πόζας ανά χρόνο . . . . .	24
2.15	Μέθοδος εκτίμησης πόζας βασισμένη στον εντοπισμό . . . . .	26
2.16	Αρχιτεκτονική simple baseline για την εκτίμηση πόζας . . . . .	27
2.17	Lifting from the deep: Παράδειγμα αρχιτεκτονικής ανύψωσης 2D σε 3D . . . . .	28
2.18	Ordinal Depth Supervision: Παράδειγμα αρχιτεκτονικής ανύψωσης 2D σε 3D . . . . .	29

# Κατάλογος πινάκων

# Ακρωνύμια Εγγράφου

Παρακάτω παρατίθενται ορισμένα από τα πιο συχνά χρησιμοποιούμενα ακρωνύμια της παρούσας διπλωματικής εργασίας:

MDE	→	Model-Driven Engineering
DSL	→	Domain Specific Language
M2M	→	Model to Model transformation
M2T	→	Model to Text transformation
IoT	→	Internet of Things
RTOS	→	Real Time Operating System
MQTT	→	Message Queuing Telemetry Transport



# 1

## Εισαγωγή

Αν και υπήρχε ως ιδέα εδώ και περίπου 50 χρόνια, το διαδίκτυο των πραγμάτων, και ως όρος αλλά και ως προς τη χρήση του, ήρθε στο επίκεντρο του ενδιαφέροντος τα τελευταία 10 χρόνια και καθημερινά γίνεται ολοένα και πιο διαδεδομένο. Πλέον μάλιστα ο συνολικός αριθμός συνδεδεμένων συσκευών στο διαδίκτυο είναι μεγαλύτερος από αυτόν του πληθυσμού της Γης<sup>1</sup>.

Το IoT βρίσκει εφαρμογή στην ανάπτυξη έξυπνων υποδομών και αυτοματοποίησης διαδικασιών. Από την δημιουργία ενός "Έξυπνου Σπιτιού" μέχρι και σε κάτι τόσο ουσιώδες όπως την καλύτερη παρακολούθηση ασθενών σε νοσοκομεία και άρα την πιο σωστή περίθαλψή τους. Άλλα παραδείγματα εφαρμογής είναι οι "Έξυπνες πόλεις" (διαχείριση κυκλοφορίας στους δρόμους, διαχείριση απορριμάτων, διανομή νερού κ.α.), τα αυτοκινούμενα οχήματα, οι αυτοματισμοί στη γεωργία, και γενικότερα στη βιομηχανία.

Η ολοένα και μεγαλύτερη διάδοση του IoT, συνεπάγεται και την αξιοποίησή του από μεγαλύτερο κοινό, στο οποίο ανήκουν και άτομα τα οποία μπορεί να γνωρίζουν σε βάθος ένα συγκεκριμένο αντικείμενο στο οποίο βρίσκει εφαρμογή το IoT, όμως δεν έχουν επαρκείς, ή και καθόλου γνώσεις προγραμματισμού (οι λεγόμενοι citizen developers). Κρίνεται σκόπιμη λοιπόν η ανάπτυξη μεθόδων που θα μετατρέπουν την δημιουργία ενός συστήματος IoT σε διαδικασία πιο φιλική προς τα άτομα αυτά. Ταυτόχρονα, καθώς η αυτοματοποίηση διαδικασιών φαίνεται να χρησιμεύει σε όλο και περισσότερους τομείς, έχει αρχίσει να εμφανίζεται η ανάπτυξη χαμηλού-κώδικα (*low-code development*), η οποία αποσκοπεί σε όσο το δυνατό λιγότερη χρήση κώδικα για την παραγωγή λογισμικού.

Εδώ βρίσκει άμεση εφαρμογή η Μοντελοστρεφής Μηχανική, η οποία προσφέ-

---

<sup>1</sup><https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>

ρει γρήγορη και πιο αυτοματοποιημένη ανάπτυξη λογισμικού. Στο πλαίσιο αυτής, οι γλώσσες συγκεκριμένου πεδίου προσφέρουν ένα πιο αφαιρετικό επίπεδο για την παραγωγή λογισμικού, επομένως αποτελούν ένα πολύ σημαντικό εργαλείο για προγραμματιστές και μη, είτε για να απλοποιηθεί η διαδικασία παραγωγής για τους πρώτους, είτε για να καλυφθεί το κενό προγραμματιστικών γνώσεων για τους δευτέρους.

### 1.1 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

---

Πέρα από τα προβλήματα που αναλύθηκαν στην προηγούμενη παράγραφο, στα οποία δίνει λύση η MDE, ένα ακόμη σημαντικό θέμα που εμφανίζεται με την ανάπτυξη του κλάδου του IoT είναι η κατασκευή όλο και περισσότερων διαφορετικών IoT συσκευών. Φυσικά, λόγω αυτού από τη μία επεκτείνονται οι δυνατότητες που ένα IoT σύστημα μπορεί να έχει, από την άλλη όμως αυξάνεται η πολυπλοκότητα και ετερογένεια στο IoT.

Υπάρχει πληθώρα IoT συσκευών στην αγορά, όπως π.χ. τα έξυπνα ρολόγια, που διανέμονται έτοιμες για χρήση. Σε αυτές τις περιπτώσεις, οι χρήστες μπορούν να ακολουθήσουν σαφείς οδηγίες χρήσης από τον κατασκευαστή, και άρα πολύ εύκολα να αξιοποιήσουν τις δυνατότητες που η εκάστοτε συσκευή προσφέρει. Επομένως, το πρόβλημα της πολυπλοκότητας δεν εμφανίζεται σε τέτοιου είδους IoT συσκευές.

Στην περίπτωση όμως που κάποιος/α επιθυμεί να αναπτύξει ένα σύστημα με IoT συσκευές από την αρχή, επειδή π.χ. θέλει να πειραματιστεί ή να υλοποιήσει κάποιες λειτουργίες πιο εξειδικευμένες, τότε απαιτείται μια μεγάλη διαδικασία για την κατασκευή και άρα πολλές γνώσεις. Το πρώτο βήμα είναι η επιλογή των κατάλληλων μικροελεγκτών, αισθητήρων, ενεργοποιητών για την υλοποίηση της ιδέας. Απαιτείται λοιπόν γνώση πάνω στον τρόπο λειτουργίας των συσκευών αυτών, καθώς και στον τρόπο διασύνδεσης και επικοινωνίας τους. Ακολουθεί η ανάπτυξη λογισμικού για την υλοποίηση των επιθυμητών λειτουργιών, κάτι το οποίο από μόνο του σημαίνει πως πρέπει να υπάρχει εμπειρία με προγραμματισμό και πρωτόκολλα επικοινωνίας. Επίσης, σε πολλές περιπτώσεις, στο σύστημα που υλοποιείται απαιτείται η ύπαρξη ιδιοτήτων όπως η ακρίβεια στον χρόνο απόκρισης ή η χαμηλή κατανάλωση ενέργειας. Επομένως, απαιτούνται και οι γνώσεις των ιδιοτήτων των RTOS, καθώς και της κατάλληλης χρήσης τους.

### 1.2 ΣΚΟΠΟΣ - ΣΥΝΕΙΣΦΟΡΑ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

---

Η παρούσα διπλωματική έχει ως στόχο την ανάπτυξη μιας μηχανής λογισμικού μοντελοστρεφούς λογικής, με την οποία οι χρήστες θα μπορούν να μοντελοποιούν συσκευές καθώς και την διασύνδεσή τους. Οι συσκευές αυτές θα μπορούν να είναι



είτε μικροελεγκτές, είτε περιφερειακά (αισθητήρες και ενεργοποιητές), και όλα μαζί θα συνδέονται κατάλληλα για να συνθέσουν ένα σύστημα.

Αρχικά υλοποιήθηκαν δύο DSL για την περιγραφή των συσκευών και των μεταξύ τους συνδέσεων. Στην μία περιγράφονται τα χαρακτηριστικά των συσκευών (μνήμη, μονάδα επεξεργασίας, δικτύωση, ακροδέκτες κ.α.) και στην άλλη οι μεταξύ τους συνδέσεις (συνδέσεις ακροδεκτών, πρωτόκολλα επικοινωνίας που χρησιμοποιούνται κ.α.). Μέσω αυτών, δημιουργούνται τα κατάλληλα μοντέλα για τις συσκευές και συνδέσεις.

Από τα μοντέλα, πραγματοποιούνται δύο μετασχηματισμοί, ένας Model-to-Text (M2T) και ένας Model-to-Model (M2M). Ο M2M έχει ως αποτέλεσμα την παραγωγή διαγραμμάτων, τα οποία βοηθούν στην οπτικοποίηση της συνδεσμολογίας και ενδοεπικοινωνίας του συστήματος. Μέσω του M2T, παράγονται αυτόματα τμήματα λογισμικού που θα υλοποιούν κάποιες βασικές λειτουργίες (λήψη μετρήσεων από αισθητήρες, έλεγχος ενεργοποιητών κ.α.). Ο παραγόμενος κώδικας θα αφορά συσκευές που υποστηρίζονται από το λειτουργικό σύστημα πραγματικού χρόνου RIOT.

---

## 1.3 ΔΙΑΦΘΡΩΣΗ ΤΗΣ ΑΝΑΦΟΡΑΣ

Η διάρθρωση της παρούσας διπλωματικής εργασίας είναι η εξής:

- **Κεφάλαιο ??:** Γίνεται ανασκόπηση της ερευνητικής δραστηριότητας στον τομέα μέχρι σήμερα.
- **Κεφάλαιο 2:** Αναλύεται το θεωρητικό υπόβαθρο.
- **Κεφάλαιο ??:** Παρουσιάζονται οι διάφορες τεχνικές και τα εργαλεία που χρησιμοποιήθηκαν στις υλοποιήσεις.
- **Κεφάλαιο ??:** Παρουσιάζονται τα βήματα της μεθοδολογίας που υλοποιήθηκε.
- **Κεφάλαιο ??:** Περιγράφονται 3 παραδείγματα εφαρμογής των εργαλείων που αναπτύχθηκαν.
- **Κεφάλαιο ??:** Παρουσιάζονται τα τελικά συμπεράσματα και προτείνονται θέματα για μελλοντική μελέτη, αλλαγές και επεκτάσεις.

# 2

## Θεωρητικό Υπόβαθρο - Αρχιτεκτονική

### 2.1 ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

---

Σύμφωνα με το [2]: Τα **Λειτουργικά Συστήματα** (εν συντομία **ΛΣ**) είναι προγράμματα που ελέγχουν την εκτέλεση προγραμμάτων εφαρμογών και δρουν ως διεπαφή ανάμεσα στις εφαρμογές και το υλικό του υπολογιστή. Θα λέγαμε ότι έχουν τρεις στόχους:

- **Ευκολία:** Τα ΛΣ κάνουν ευκολότερη τη χρήση ενός υπολογιστή.
- **Αποτελεσματικότητα:** Τα ΛΣ επιτρέπουν την αποτελεσματική χρήση των πόρων ενός υπολογιστικού συστήματος.
- **Ικανότητα Εξέλιξης:** Τα ΛΣ πρέπει να είναι κατασκευασμένα με τέτοιο τρόπο, ώστε να επιτρέπουν την αποτελεσματική ανάπτυξη, τον έλεγχο και την εισαγωγή νέων λειτουργιών συστήματος, χωρίς να παρεμβαίνουν στην παροχή υπηρεσιών.

#### 2.1.1 Το ΛΣ ως Διεπαφή Χρήστη/Υπολογιστή

Ένας υπολογιστής είναι ένα σύστημα που αποτελείται από το υλικό (hardware) και το λογισμικό (software). Το υλικό είναι ο φυσικός εξοπλισμός (οθόνες, μνήμες, εκτυπωτές κτλ) ενώ το λογισμικό είναι μια συλλογή προγραμμάτων που επιτρέπουν στο υλικό να λειτουργεί [3]. Το λογισμικό διαιρείται σε 2 κατηγορίες: στα προγράμματα εφαρμογών και στα λειτουργικά συστήματα. Το υλικό και το λογι-



Σχήμα 2.1: Δομή Υλικού και Λογισμικού Υπολογιστή

σμικό που αποτελούν ένα υπολογιστικό σύστημα μπορεί να αναπαρασταθεί από μια ιεραρχική δομή ή μια δομή επιπέδων, όπως φαίνεται στο Σχήμα 2.1.

Το ΛΣ, επομένως, «κάθεται» ανάμεσα στο υλικό και στις εφαρμογές που χρησιμοποιεί ο χρήστης ο οποίος συνήθως δεν ενδιαφέρεται για λεπτομέρειες του υλικού του υπολογιστή. Δρα, δηλαδή, ως ένας διαμεσολαβητής που διευκολύνει την επικοινωνία του χρήστη με τον υπολογιστή χωρίς ο πρώτος να χρειάζεται να γνωρίζει την γλώσσα του τελευταίου, ενώ είναι υπεύθυνο για την ομαλή λειτουργία των προγραμμάτων εφαρμογών. Για παράδειγμα, προκειμένου να εμφανιστεί το κείμενο «Χαίρε Κόσμε» στην οθόνη, πρέπει μερικές εκατοντάδες πίξελ να λειτουργήσουν σε συγκεκριμένες θέσεις. Αυτό μπορεί να γίνει με το να διαβάσει κανείς τις προδιαγραφές του υλικού και να γράφει κώδικα που χειρίζεται τα σωστά μπιτς στην μνήμη κάτι που είναι αρκετά επίπονο. Οι περισσότεροι χρήστες όμως θέλουν απλά να γράφουν την εντολή `print("Hello World")` χωρίς να νοιαστούν για περαιτέρω λεπτομέρειες. Εκεί είναι που μπαίνει το ΛΣ για να δώσει λύση [4].

Πιο αναλυτικά, μερικές από τις περιοχές στις οποίες το ΛΣ παρέχει υπηρεσίες περιγράφονται παρακάτω σύμφωνα με το [2]:

- **Στην ανάπτυξη προγραμμάτων:** Το ΛΣ παρέχει πληθώρα υπηρεσιών, όπως επεξεργαστές κειμένου και διορθωτές λαθών (debuggers), προκειμένου να

βοηθήσει τον προγραμματιστή στην ανάπτυξη εφαρμογών. Συνήθως, οι υπηρεσίες αυτές έχουν τη μορφή βοηθητικών προγραμμάτων, τα οποία αν και δεν αποτελούν αυστηρά τμήμα του πυρήνα του λειτουργικού συστήματος, παρέχονται μαζί με το ΛΣ και αναφέρονται ως εργαλεία ανάπτυξης προγραμμάτων εφαρμογών.

- **Στην εκτέλεση προγραμμάτων:** Για την εκτέλεση ενός προγράμματος απαιτείται πλήθος βημάτων. Εντολές και δεδομένα πρέπει να φορτωθούν στην κύρια μνήμη, συσκευές Εισόδου/Εξόδου (E/E) και αρχεία πρέπει να αρχικοποιηθούν και διάφοροι άλλοι πόροι πρέπει να ετοιμαστούν. Το ΛΣ διαχειρίζεται αυτές τις υποχρεώσεις χρονοδρομολόγησης για λογαριασμό του χρήστη.
- **Στην πρόσβαση σε συσκευές Εισόδου/Εξόδου(E/E):** Κάθε συσκευή E/E απαιτεί το δικό της ιδιαίτερο σύνολο εντολών ή σημάτων ελέγχου για τη λειτουργία της. Το ΛΣ παρέχει ενιαία διεπαφή που αποκρύπτει αυτές τις λεπτομέρειες, έτσι ώστε οι προγραμματιστές να έχουν πρόσβαση σε τέτοιες συσκευές χρησιμοποιώντας απλές αναγνώσεις (reads) και εγγραφές (writes).
- **Στην ελεγχόμενη πρόσβαση σε αρχεία:** Το ΛΣ πρέπει να αναπαριστά τη λεπτομερή κατανόηση, όχι μόνο της φύσης των συσκευών E/E (οδηγού δίσκου, οδηγού ταινίας), αλλά και της δομής των δεδομένων που βρίσκονται σε αρχεία στο αποθηκευτικό μέσο. Επίσης, στην περίπτωση συστημάτων πολλαπλών χρηστών, το ΛΣ μπορεί να παρέχει μηχανισμούς προστασίας για τον έλεγχο της πρόσβασης σε αρχεία.
- **Στην πρόσβαση στο σύστημα:** Σε ό,τι αφορά διαμοιραζόμενα ή δημόσια συστήματα, το ΛΣ ελέγχει συνολικά την πρόσβαση στο σύστημα και σε συγκεκριμένους πόρους του. Η λειτουργία πρόσβασης πρέπει να παρέχει προστασία των πόρων και των δεδομένων από μη εξουσιοδοτημένους χρήστες και να διευθετεί θέματα συγκρούσεων και διεκδίκησης πόρων.
- **Στην ανίχνευση σφαλμάτων και στην απόκριση:** Διάφορα σφάλματα μπορούν να προκύψουν κατά τη διάρκεια λειτουργίας ενός υπολογιστικού συστήματος. Μεταξύ αυτών περιλαμβάνονται εσωτερικά και εξωτερικά σφάλματα υλικού, όπως σφάλματα μνήμης ή δυσλειτουργία κάποιας συσκευής, καθώς και διάφορα σφάλματα λογισμικού, όπως η διαίρεση με το μηδέν και η προσπάθεια προσπέλασης απαγορευμένης θέσης μνήμης. Σε κάθε περίπτωση, το ΛΣ πρέπει να παρέχει μια απόκριση, η οποία απαλείφει τη συνθήκη σφάλματος, με το μικρότερο δυνατό αντίκτυπο στις εφαρμογές που εκείνη την ώρα εκτελούνται. Απόκριση μπορεί να αποτελεί η λήξη του προγράμματος που προκάλεσε το σφάλμα, η επανέναρξη λειτουργίας ή ακόμα και η απλή αναφορά σφάλματος στην εφαρμογή.
- **Στην λογιστική:** Ένα καλό ΛΣ συλλέγει στατιστικά χρήσης διάφορων πόρων και παρακολουθεί παραμέτρους απόδοσης, όπως είναι ο χρόνος απόκρισης. Σε κάθε σύστημα, οι πληροφορίες αυτές είναι χρήσιμες για τη λήψη αποφάσεων σχετικών με μελλοντικές αναβαθμίσεις και με τις ρυθμίσεις του συστήματος, ώστε να επιτυγχάνεται η βελτίωση της απόδοσης. Σε συστήματα πολλαπλών χρηστών, οι πληροφορίες αυτές μπορούν να χρησιμοποιηθούν και για λόγους χρέωσης.

### 2.1.2 Το ΛΣ ως Διαχειριστής Πόρων

Όπως αναφέρεται και παραπάνω το ΛΣ είναι μεταξύ άλλων υπεύθυνο για τον έλεγχο των πόρων του υπολογιστή, δηλαδή τις μνήμες, τους δίσκους, τον επεξεργαστή κτλ. Αναφέρθηκε, επίσης, ότι το ΛΣ είναι και αυτό ένα λογισμικό και άρα λειτουργεί με τον ίδιο τρόπο που λειτουργεί οποιαδήποτε άλλη εφαρμογή, δηλαδή είναι μια ακολουθία εντολών που εκτελούνται από τον επεξεργαστή. Κατά την φάση εκτέλεσης, το ΛΣ κατανέμει τον χρόνο του επεξεργαστή και τους πόρους του υπολογιστή στα προγράμματα που πρόκειται να τρέξουν. Για να εκτελεστεί όμως μια εφαρμογή ή ένα πρόγραμμα ο επεξεργαστής πρέπει να διακόψει την εκτέλεση του ΛΣ, ενώ μετά το πέρας των εντολών επαναφέρει το ΛΣ στον έλεγχο προκειμένου να γίνουν οι απαραίτητες προετοιμασίες για τις επόμενες εργασίες [2].

Ακριβώς επειδή το ΛΣ βοηθάει την εκτέλεση σχεδόν κάθε προγράμματος πρέπει να είναι και αρκετά αποδοτικό. Για παράδειγμα, τα προγράμματα εφαρμογών δημιουργούν αντικείμενα και πίνακες συνέχεια οπότε είναι σημαντικό αυτή η δουλειά να γίνεται γρήγορα και με εξοικονόμηση πόρων. Οποιοδήποτε κέρδος του ΛΣ, είτε σε ταχύτητα είτε σε μνήμη, μπορεί να επηρεάσει δραματικά την απόδοση του υπολογιστικού συστήματος [4].

## 2.2 OPEN GRAPHICS LIBRARY

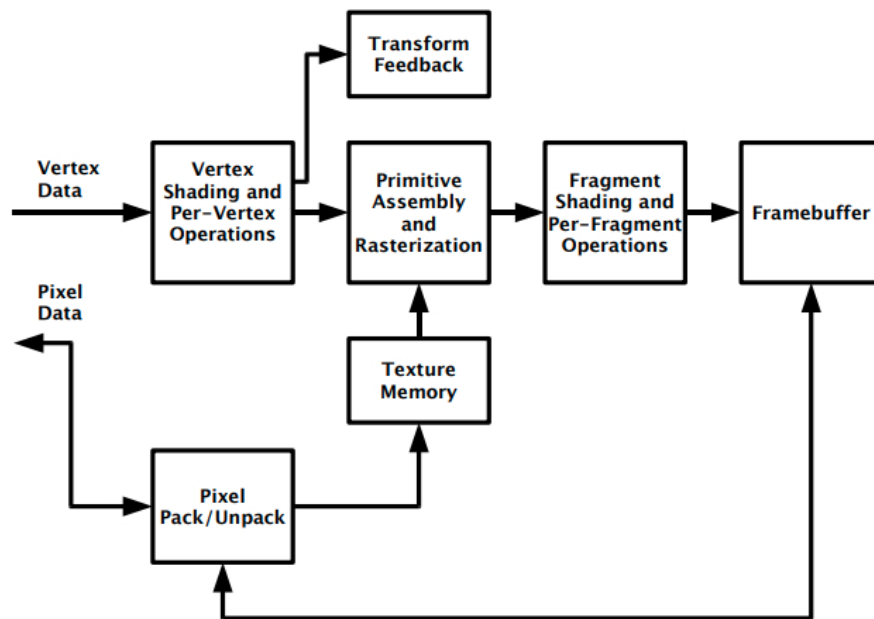
---

Αντίστοιχα με το ΛΣ αλλά ανεξάρτητο από αυτό, η **Open Graphics Library**[5] (εν συντομία **OpenGL**) αποτελεί μία *Διεπαφή Προγραμματισμού Εφαρμογών* για τη βέλτιστη χρήση των πόρων της κάρτας γραφικών με σκοπό την οικονομική και γρήγορη απόδοση 2D και 3D γραφικών. Η διεπαφή περιλαμβάνει ένα σύνολο εντολών οι οποίες επιτρέπουν στον χρήστη να καθορίσει τα απαραίτητα αντικείμενα και διεργασίες για την παραγωγή υψηλής ποιότητας έγχρωμων εικόνων δισδιάστατων ή τρισδιάστατων αντικειμένων.

### 2.2.1 OpenGL

Η OpenGL είναι υπεύθυνη για την επεξεργασία των δεδομένων στην μνήμη της κάρτας γραφικών, την εγγραφή δεδομένων στον framebuffer και την ανάγνωση του. Ο framebuffer απαρτίζεται από ένα σύνολο pixels διατεταγμένων σε ένα δισδιάστατο πίνακα. Κάθε στοιχείο του πίνακα αποτελείται από έναν αριθμό bits, ανάλογα με την υλοποίηση της OpenGL, τα οποία καθορίζουν το χρώμα, το βάθος και το στένσιλ για κάθε pixel.

Στο Σχήμα 2.2 φαίνεται η διαδικασία απεικόνισης γραφικών της OpenGL. Αρχικά, δέχεται ως είσοδο τα δεδομένα των κορυφών των αντικειμένων προς προβολή,



Σχήμα 2.2: Διαδικασία απεικόνισης γραφικών της OpenGL

τα οποία συνθέτονται σε πρωτόγονα σχήματα όπως κορυφές, τμήματα γραμμών, επιφάνειες και πολύγωνα. Στη συνέχεια, οι κορυφές μεταμορφώνονται σε γεωμετρικά πρωτόγονα σχήματα, συνήθως τρίγωνα ή πολύγωνα, τα οποία μέσω ψηφίδωσης μπορούν να παράξουν περισσότερα πρωτόγονα σχήματα από μία είσοδο. Προαιρετικά, τα αποτελέσματα από αυτά τα στάδια δύναται να ανατροφοδοτήσουν ενδιάμεσες μνήμες για μετέπειτα χρήση.

Τα τελικά πρωτόγονα σχήματα περικόπτονται από έναν καθορισμένο όγκο για να προετοιμαστούν για το στάδιο της ψηφιοποίησης. Η διαδικασία αυτή παράγει σειρές από διευθύνσεις του framebuffer συνοδευόμενες από τιμές που περιγράφουν τη δισδιάστατη απεικόνιση των αρχικών τρισδιάστατων πρωτόγονων σχημάτων. Κάθε τμήμα που παράγεται με αυτό τον τρόπο υπόκειται σε περαιτέρω διεργασίες ξεχωριστά. Οι διεργασίες περιλαμβάνουν υπό συνθήκη ενημέρωση των τιμών σε συνάρτηση με εισερχόμενες ή αποθηκευμένες τιμές του βάθους ή του στένσιλ, ανάμειξη των εισερχόμενων χρωμάτων με αποθηκευμένα χρώματα ή άλλες λογικές πράξεις στις τιμές των τμημάτων. Τέλος, ο framebuffer επικαιροποιείται με τις τιμές των τμημάτων, προβάλλοντας έτσι την τελική εικόνα στην οθόνη.

## 2.3 ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ

Το **Διαδίκτυο των Πραγμάτων** (μτφ. **Internet of Things**, εν συντομία **IoT**) είναι ένα σύνολο φυσικών αντικειμένων τα οποία έχουν αισθητήρες, λογισμικό, υπολογιστική ισχύ κτλ. και τα οποία ανταλλάσσουν δεδομένα μεταξύ τους μέσω του

Διαδικτύου ή ενός οποιουδήποτε άλλου δικτύου επικοινωνίας. Τα αντικείμενα αυτά μπορεί να είναι ένας άνθρωπος με βηματοδότη στην καρδιά, ένα έξυπνο ρολόι που μετράει παλμούς, ένας καθρέφτης που διαβάζει δεδομένα από αισθητήρες και εμφανίζει πληροφορίες για την κατάσταση του χρήστη.

### 2.3.1 Πώς δουλεύει το IoT

Ένα σύστημα IoT περιέχει πολλές φυσικές μονάδες (hardware) οι οποίες μπορούν να χωριστούν στις ακόλουθες κατηγορίες [6]:

- Αισθητήρες & Ενεργοποιητές (Sensors & Actuators)
- Μονάδες Επεξεργασίας (Processing Units)
- Μονάδες Αποθήκευσης (Storage Units)
- Μονάδες Επικοινωνίας (Communication Units)

Σε ό,τι αφορά το λογισμικό των IoT συσκευών, δεν υπάρχει κάποια συγκεκριμένη αρχιτεκτονική που να ακολουθείται καθολικά. Έχουν προταθεί αρκετές, ενώ παρακάτω παρουσιάζονται 2 από τις πιο διαδεδομένες, η αρχιτεκτονική 3 επιπέδων και η αρχιτεκτονική 5 επιπέδων [7].

#### Αρχιτεκτονική 3 επιπέδων

Η αρχιτεκτονική 3 επιπέδων είναι από τις πιο βασικές και πρώιμες αρχιτεκτονικές που αναπτύχθηκαν και απεικονίζεται στο Σχήμα 2.3.

Όπως προκύπτει και από το όνομα αποτελείται από 3 επίπεδα:

- **Επίπεδο Αντίληψης:** Το επίπεδο αντίληψης είναι το φυσικό επίπεδο (hardware) το οποίο περιέχει αισθητήρες για λήψη πληροφοριών και ανίχνευση παραμέτρων από τον περιβάλλοντα χώρο.
- **Επίπεδο Δικτύου:** Το επίπεδο δικτύου είναι υπεύθυνο για την διασύνδεση με άλλες συσκευές, έξυπνα πράγματα κτλ. Χρησιμοποιείται επίσης για την μετάδοση και επεξεργασία δεδομένων των αισθητήρων.
- **Επίπεδο Εφαρμογής:** Το επίπεδο εφαρμογής είναι υπεύθυνο για την παροχή υπηρεσιών συγκεκριμένης εφαρμογής (application specific services) στον χρήστη. Καθορίζει τις διάφορες εφαρμογές στις οποίες μπορεί να αναπτυχθεί το IoT, όπως το έξυπνο σπίτι, ο έξυπνος καθρέφτης κ.α.

#### Αρχιτεκτονική 5 επιπέδων

Η αρχιτεκτονική των 3 επιπέδων περιγράφει την βασική ιδέα πίσω από το IoT, αλλά στην πράξη δεν επαρκή. Για το λόγο αυτό, υπάρχουν αρχιτεκτονικές με πε-





Σχήμα 2.3: Αρχιτεκτονική 3 Επιπέδων

ρισσότερα επίπεδα στην βιβλιογραφία. Μία από αυτές είναι η αρχιτεκτονική 5 επιπέδων που φαίνεται στο Σχήμα 2.4.

Στην περίπτωση αυτή, τα επίπεδα Αντίληψης και Εφαρμογής παραμένουν ίδια με αυτά της αρχιτεκτονικής 3 επιπέδων ενώ επεξηγούνται και τα υπόλοιπα 3 επίπεδα:

- **Επίπεδο Μεταφοράς:** Το επίπεδο μεταφοράς, μεταδίδει τα δεδομένα των αισθητήρων από το επίπεδο αντίληψης στο επίπεδο επεξεργασίας και αντίστροφα μέσω δικτύων όπως 3G, LAN, RFID, NFC κτλ.
- **Επίπεδο Επεξεργασίας:** Το επίπεδο επεξεργασίας, γνωστό και ως middleware, αποθηκεύει, αναλύει και επεξεργάζεται μεγάλο όγκο δεδομένων που προέρχονται από το επίπεδο μεταφοράς. Προσφέρει μια ευρεία γκάμα υπηρεσιών στα χαμηλότερα επίπεδα. Χρησιμοποιεί επίσης πολλές τεχνολογίες όπως βάσεις δεδομένων, υπολογιστική νέφους και ενότητες επεξεργασίας μεγάλων δεδομένων.
- **Επίπεδο Επιχείρησης:** Το επίπεδο επιχείρησης διαχειρίζεται όλο το IoT σύστημα, συμπεριλαμβανομένων των εφαρμογών του, των επιχειρηματικών του μοντέλων και της ιδιωτικότητας των χρηστών.



**Επίπεδο Επιχείρησης  
(Business Layer)**

**Επίπεδο Εφαρμογής  
(Application Layer)**

**Επίπεδο  
Επεξεργασίας  
(Processing Layer)**

**Επίπεδο Μεταφοράς  
(Transport Layer)**

**Επίπεδο Αντίληψης  
(Perception Layer)**

Σχήμα 2.4: Αρχιτεκτονική 5 Επιπέδων

---

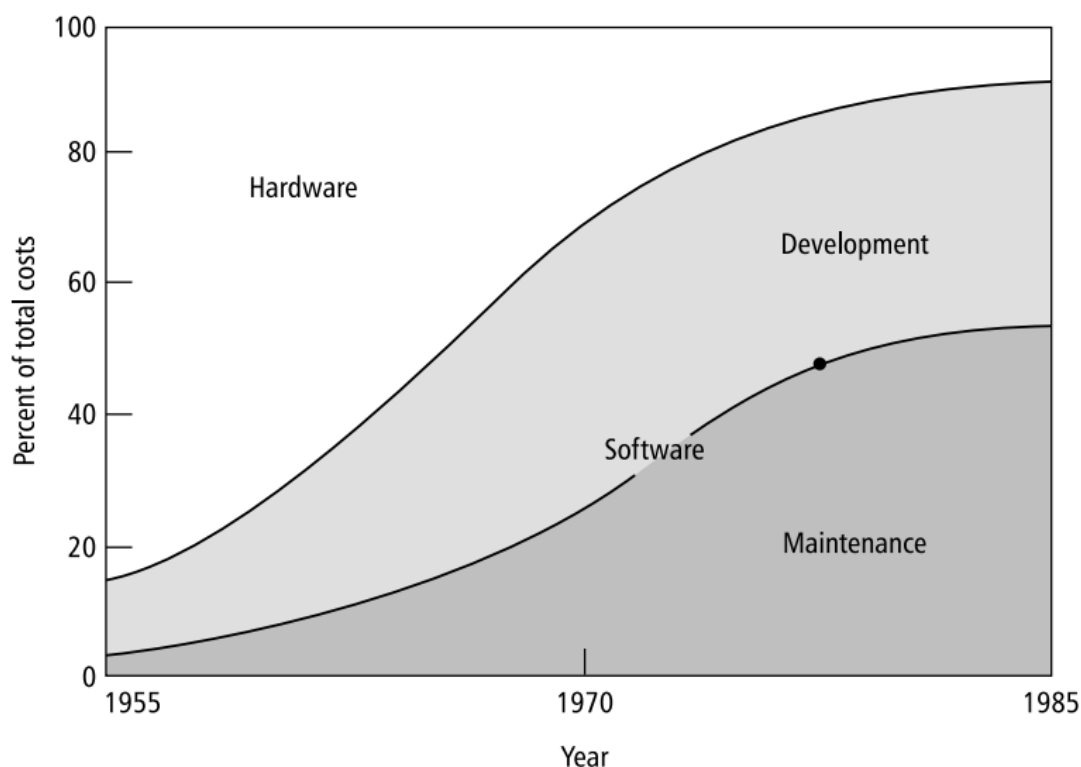
## 2.4 ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ

---

Στη σύγχρονη πραγματικότητα, το λογισμικό έχει κυριαρχήσει στην καθημερινή ζωή, καθιστώντας το μία από τις σημαντικότερες τεχνολογίες. Παίζοντας ταυτόχρονα το ρόλο προϊόντος και εργαλείου, αποτελεί τη βάση της επιστημονικής έρευνας και της επίλυσης προβλημάτων μηχανικής, καθιστώντας δυνατή την δημιουργία καινούργιων και την επέκταση υπαρχουσών τεχνολογιών (π.χ. γενετική μηχανική και τηλεπικοινωνίες αντίστοιχα). Ταυτόχρονα, έχει διεισδύσει σε συστήματα κάθε είδους: μεταφοράς, ιατρικά, τηλεπικοινωνιών, στρατιωτικά, βιομηχανικά, ψυχαγωγίας κ.α. αλλάζοντας τον τρόπο που αντιλαμβανόμαστε και αλληλεπιδράμε με τον κόσμο.

Έτσι, τα λογισμικά προγράμματα αντιμετωπίζουν ολοένα και περισσότερα προβλήματα της καθημερινής ζωής, αυξάνοντας την αναγκαιότητα και το κόστος τους. Πράγματι, το συνολικό κόστος λογισμικού υπολογίζεται στα €500 δισεκατομμύρια

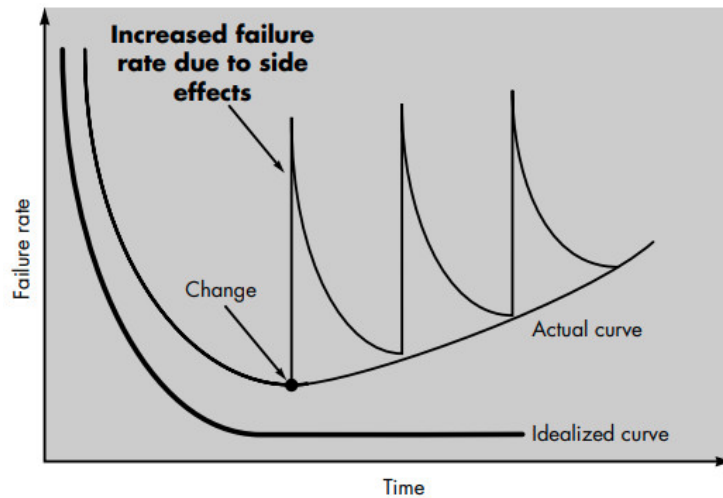
στην Αμερική και διπλάσιο παγκοσμίως [8]. Αυτό αναφέρεται τόσο στο κόστος ανάπτυξης του λογισμικού όσο και συντήρησής του αφού έχει παραδοθεί στον πελάτη. Ταυτόχρονα, το κόστος του υλικού έχει μειωθεί δραματικά, αποτελώντας λιγότερο από το 20% των συνολικών εξόδων ενός συστήματος όπως φαίνεται στο Σχήμα 2.5.



Σχήμα 2.5: Σχετική κατανομή κόστους υλικού/λογισμικού (Πηγή: [1])

Η διαφορά κόστους έγκειται στο γεγονός ότι τα σύγχρονα προγράμματα λογισμικού είναι μεγάλα και περίπλοκα, απαιτώντας ομάδες υψηλής εξειδίκευσης, δεν έχουν περιορισμούς (δηλαδή έχουν περισσότερους βαθμούς ελευθερίας) και τέλος επειδή υφίσταται συνεχόμενες αλλαγές. Μάλιστα, στη διάρκεια ζωής του λογισμικού οι αλλαγές αυτές ενδέχεται να εισάγουν σφάλματα, αυξάνοντας τον κίνδυνο αποτυχίας του συστήματος, όπως φαίνεται στο Σχήμα 2.6. Επιπλέον, σε αντίθεση με την αποτυχία του υλικού, όπου αντιμετωπίζεται με αντικατάσταση του χαλασμένου μέρους, η αποτυχία του λογισμικού έγκειται σε σφάλμα σχεδιασμού, καθιστώντας τη συντήρηση του σημαντικά πιο περίπλοκη και ακριβή διαδικασία.

Εντούτοις, η αυξανόμενη εξάρτηση των καθημερινών δραστηριοτήτων από περίπλοκα συστήματα λογισμικού απαιτεί την εύρωστη και αλάνθαστη λειτουργία τους καθ' όλη τη διάρκεια ζωής του λογισμικού, το οποίο όμως αλλάζει συνεχώς αυξάνοντας τον κίνδυνο αποτυχίας του. Τοιουτοτρόπως, για την μείωση του ρίσκου αποτυχίας και την απλούστευση της ανάπτυξης και συντήρησης υψηλής ποιότητας έργων λογισμικού, έπρεπε να τυποποιηθούν οι μέθοδοι και διαδικασίες που διέπουν ολόκληρο τον κύκλο ζωής του. Αυτές οι μεθοδολογίες εμπεριέχονται και αναλύονται από την επιστήμη της τεχνολογίας λογισμικού.



Σχήμα 2.6: Καμπύλη αποτυχίας για το λογισμικό

### 2.4.1 Τεχνολογία Λογισμικού

Η τεχνολογία λογισμικού είναι η συστηματική, πειθαρχημένη και μετρήσιμη προσέγγιση με σκοπό την ανάπτυξη, λειτουργία και συντήρηση υψηλής ποιότητας έργων λογισμικού. Οι διαδικασίες της τεχνολογίας λογισμικού παρέχουν ένα ευρύτερο πλαίσιο, μέσα στο οποίο εφαρμόζονται οι τεχνικές μέθοδοι, παράγονται προϊόντα δουλειάς (μοντέλα, έγγραφα, δεδομένα, αναφορές κ.α.), εξακριβώνονται σημεία σταθμοί, εξασφαλίζεται η ποιότητα και γίνεται κατάλληλη διαχείριση των αλλαγών των απαιτήσεων.

Οι μέθοδοι της τεχνολογίας λογισμικού είναι το τεχνικό εγχειρίδιο για την ανάπτυξη λογισμικού. Οι μέθοδοι εγχολπώνουν ένα ευρύ σύνολο λειτουργιών συμπεριλαμβανομένων την ανάλυση απαιτήσεων, σχεδιασμού, δημιουργία προγράμματος, δοκιμής και υποστήριξης. Οι μέθοδοι της τεχνολογίας λογισμικού βασίζονται σε βασικές αρχές και πρακτικές που διέπουν κάθε τομέα της τεχνολογίας και περιλαμβάνουν δραστηριότητες μοντελοποίησης και άλλες περιγραφικές τεχνικές.

#### Πλαίσιο διαδικασίας

Το πλαίσιο διαδικασίας της τεχνολογίας λογισμικού απαρτίζεται από δραστηριότητες που είναι κοινές για οποιοδήποτε έργο λογισμικού ανεξάρτητα από το μέγεθος και την περιπλοκότητα του. Ένα γενικό πλαίσιο διαδικασίας περιλαμβάνει πέντε κύριες δραστηριότητες [9]:

- **Επικοινωνία (Communication):** Πριν την έναρξη οποιασδήποτε τεχνικής εργασίας, είναι άκρως σημαντική η επικοινωνία και συνεργασία με τον πελάτη (και άλλα ενδιαφερόμενα μέλη). Η πρόθεση είναι η κατανόηση των στόχων των ενδιαφερόμενων μελών για το έργο λογισμικού και η συλλογή απαιτή-

σεων, βοηθώντας τον καθορισμό των λειτουργιών και των χαρακτηριστικών του λογισμικού.

- **Προγραμματισμός (Planning):** Ένα έργο λογισμικού είναι ένα περίπλοκο ταξίδι και η διαδικασία του προγραμματισμού είναι ο χάρτης ο οποίος οδηγεί την ομάδα ανάπτυξης. Το σχέδιο του έργου αποσαφηνίζει την απαιτούμενη δουλειά, περιγράφοντας τις τεχνικές εργασίες προς διεκπεραίωση, τα πιθανά ρίσκα, τους απαιτούμενους πόρους, τα προϊόντα προς παραγωγή και το πρόγραμμα εργασίας.
- **Μοντελοποίηση (Modelling):** Δημιουργούνται μοντέλα ώστε να γίνει εφικτή η καλύτερη κατανόηση των απαιτήσεων του λογισμικού και του σχεδιασμού ο οποίος δύναται να ικανοποιήσει αυτές τις απαιτήσεις. Γίνεται εμφανής η αρχιτεκτονική μορφή του έργου, ο τρόπος αλληλεξάρτησης των μεμονωμένων μερών και πολλά άλλα χαρακτηριστικά, στη προσπάθεια κατανόησης του προβλήματος και του τρόπου επίλυσης του.
- **Κατασκευή (Construction):** Η δραστηριότητα αυτή περιλαμβάνει την υλοποίηση του έργου παράγοντας κώδικα (είτε με χειροκίνητο ή αυτόματο τρόπο) και τον έλεγχο του κώδικα για την ανίχνευση σφαλμάτων.
- **Εγκατάσταση (Deployment):** Το λογισμικό ως ολόκληρη οντότητα ή σε επιμέρους διαδοχικά στάδια παραδίδεται στον πελάτη ο οποίος αξιολογεί το παραδοθέν προϊόν και παρέχει ανατροφοδότηση.

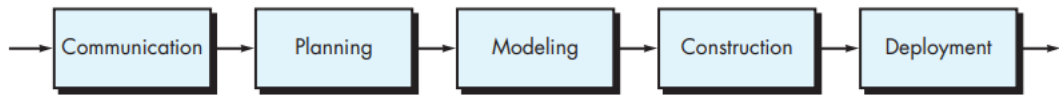
Σε πολλά έργα λογισμικού, οι δραστηριότητες πλαισίου εφαρμόζονται επαναληπτικά καθώς αναπτύσσεται το έργο. Κάθε επανάληψη παράγει μία επαύξηση του λογισμικού η οποία θέτει στην διάθεση των ενδιαφερόμενων μελών ένα υποσύνολο των τελικών χαρακτηριστικών και λειτουργιών του λογισμικού. Τοιουτοτρόπως, το λογισμικό ολοκληρώνεται σταδιακά με το πέρας κάθε επανάληψης.

### 2.4.2 Ροές διαδικασίας Τεχνολογίας Λογισμικού

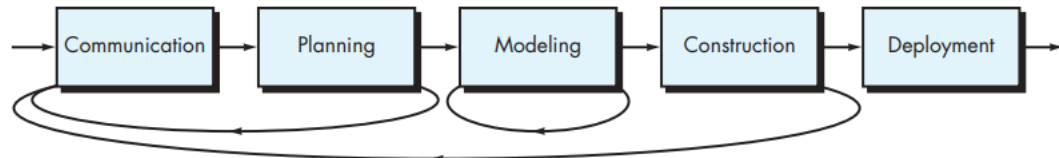
Μία διαδικασία ορίζεται ως το άθροισμα των δραστηριοτήτων, ενεργειών και εργασιών που πρέπει να διεκπεραιωθούν με σκοπό την παραγωγή ενός προϊόντος εργασίας. Όπως αναφέρθηκε παραπάνω, το γενικό πλαίσιο διαδικασίας του λογισμικού αποτελείται από πέντε δραστηριότητες σε συνδυασμό με άλλες ενέργειες - όπως παρακολούθηση και έλεγχος έργου, εκτίμηση και διαχείριση ρίσκου, διασφάλιση ποιότητας, ρύθμιση παραμέτρων κ.α. - που εφαρμόζονται καθ' όλη την διαδικασία.

Η ροή διαδικασίας περιγράφει τον τρόπο με τον οποίο οι δραστηριότητες πλαισίου, οι ενέργειες και οι εργασίες εντός κάθε δραστηριότητας οργανώνονται σε σχέση με την αλληλουχία τους και τον χρόνο όπως φαίνεται στο Σχήμα 2.7.

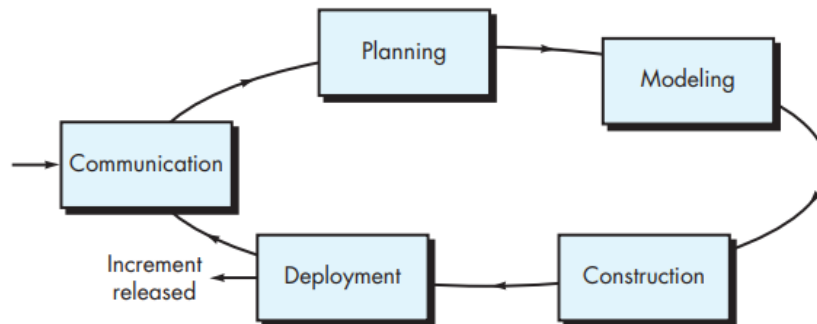
Η γραμμική ροή εκτελεί τις δραστηριότητες διαδοχικά, ξεκινώντας την κάθε δραστηριότητα με το πέρας της προηγούμενης (2.7α'). Η επαναληπτική ροή επαναλαμβάνει μία ή περισσότερες δραστηριότητες πριν προχωρήσει στην επόμενη



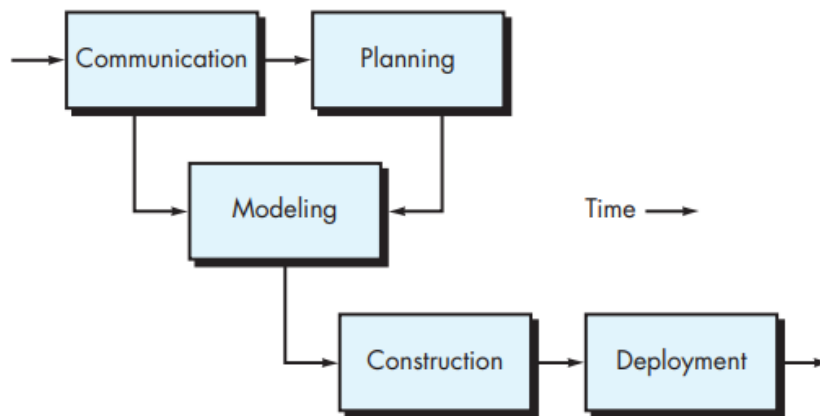
(α') Γραμμική ροή



(β') Επαναληπτική ροή



(γ') Εξελικτική ροή



(δ') Παράλληλη ροή

Σχήμα 2.7: Ροές διαδικασίας λογισμικού

(2.7β'). Η εξελικτική ροή εκτελεί τις δραστηριότητες κυκλικά. Στο πέρας της διεκπεραίωσης των δραστηριοτήτων παράγεται σταδιακά η εφαρμογή ολοένα και πιο ολοκληρωμένη (2.7γ'). Η παράλληλη ροή (2.7δ') εκτελεί μία ή περισσότερες δραστηριότητες παράλληλα με άλλες δραστηριότητες (για παράδειγμα η μοντελοποίηση μιας πτυχής του λογισμικού μπορεί να γίνεται ταυτόχρονα με την κατασκευή μιας άλλης πτυχής του).

### 2.4.3 Μοντέλα ανάπτυξης λογισμικού

#### Μοντέλο Καταρράκτη

Το μοντέλο ανάπτυξης καταρράκτη ακολουθεί την γραμμική ροή, δηλαδή οι δραστηριότητες από την επικοινωνία μέχρι την εγκατάσταση διεκπεραιώνονται διαδοχικά. Το μοντέλο αυτό επιλέγεται όταν οι απαιτήσεις του λογισμικού είναι ξεκάθαρα ορισμένες και επαρκώς σταθερές.

Το μοντέλο καταρράκτη είναι το παλαιότερο μοντέλο της τεχνολογίας λογισμικού. Εντούτοις, με την εξέλιξη της τεχνολογίας και των απαιτήσεων έχει παρουσιάσει πληθώρα προβλημάτων.

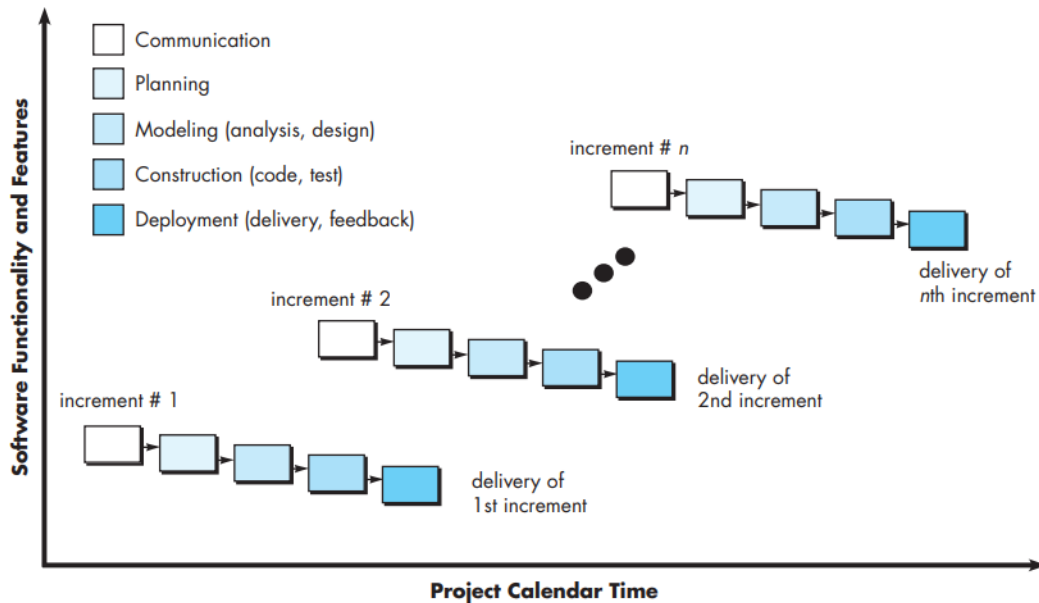
- Τα έργα λογισμικού σπάνια ακολουθούν την προτεινόμενη γραμμική ροή. Τοιουτοτρόπως, οι αλλαγές δεν είναι εύκολα διαχειρίσιμες και μπορεί να προκαλέσουν σύγχυση.
- Συνήθως είναι δύσκολο για τον πελάτη να καθορίσει όλες τις απαιτήσεις αναλυτικά. Το μοντέλο καταρράκτη όμως το απαιτεί και έτσι δεν μπορεί να διαχειριστεί την αβεβαιότητα που έγκειται στην έναρξη οποιουδήποτε έργου λογισμικού.
- Μία λειτουργική έκδοση του λογισμικού γίνεται διαθέσιμη αρκετά αργά στην διάρκεια ζωής του έργου. Έτσι, από την μία ο πελάτης πρέπει να είναι υπομονετικός και από την άλλη μία αστοχία σχεδιασμού που θα ανιχνευτεί κατά την αξιολόγηση της λειτουργικής έκδοσης μπορεί να είναι καταστροφική.

#### Αυξητικό Μοντέλο

Το αυξητικό μοντέλο συνδυάζει την γραμμική και την παράλληλη ροή διαδικασιών. Όπως φαίνεται στο Σχήμα 2.8, το αυξητικό μοντέλο χωρίζει τις λειτουργίες και τα χαρακτηριστικά του λογισμικού σε μικρές επαυξήσεις και για κάθε επαύξηση διεκπεραιώνει γραμμικά τις δραστηριότητες, στο πέρας των οποίων παραδίδεται η λειτουργία του λογισμικού.

Οι πρώτες επαυξήσεις υλοποιούν τις κυριότερες λειτουργίες του λογισμικού, ικανοποιώντας τις αρχικές απαιτήσεις. Το προϊόν παραδίδεται στον πελάτη ή υπόκειται σε λεπτομερή αξιολόγηση. Τοιουτοτρόπως, αναπτύσσεται ένα σχέδιο για τις επόμενες επαυξήσεις. Το σχέδιο αφορά τις αλλαγές του κύριου προϊόντος, με σκοπό την καλύτερη ικανοποίηση των απαιτήσεων του πελάτη, και την ανάπτυξη επιπλέον χαρακτηριστικών και λειτουργιών. Η διαδικασία επαναλαμβάνεται μετά από την παράδοση κάθε επαύξησης έως ότου ολοκληρωθεί το λογισμικό.

Το αυξητικό μοντέλο είναι ιδανικό όταν υπάρχουν κάποιες πλήρης και καλά καθορισμένες απαιτήσεις με χαλαρά συζευγμένα μέρη. Έτσι, οι αρχικές επαυξήσεις μπορούν να προγραμματιστούν και να παραλληλοποιηθούν επαρκώς.



Σχήμα 2.8: Αυξητικό Μοντέλο

### Σπειροειδές Μοντέλο

Το σπειροειδές μοντέλο είναι ένα εξελικτικό μοντέλο με κύριο γνώμονα το ρίσκο. Όπως και στο αυξητικό μοντέλο το λογισμικό παραδίδεται σε επαναλήψεις, όμως σε αντίθεση με αυτό τα βήματα δεν είναι δραστηριότητες αλλά φάσεις, με σκοπό την αντιμετώπιση του προβλήματος με το μεγαλύτερο ρίσκο να προκαλέσει αποτυχία.

Όπως φαίνεται στο Σχήμα 2.9, οι φάσεις σε κάθε επανάληψη είναι:

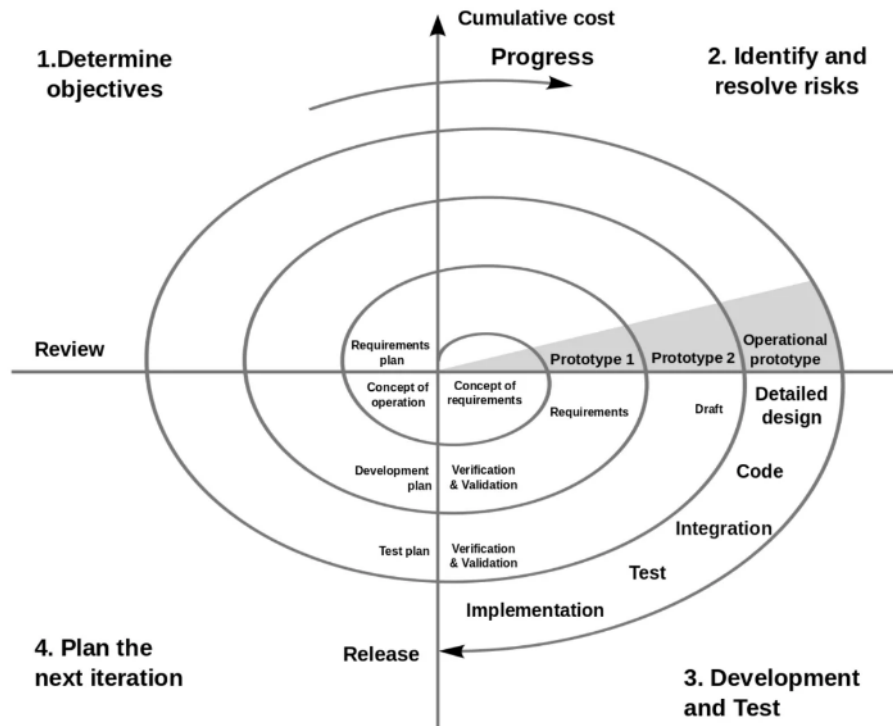
1. Εξακρίβωση του προβλήματος με το μεγαλύτερο ρίσκο, καθορισμός των στόχων και εναλλακτικών λύσεων.
2. Αξιολόγηση των εναλλακτικών και προσδιορισμός των ρίσκων, σχεδιασμός των πιθανών λύσεων.
3. Ανάπτυξη μίας λύσης, επιβεβαίωση καταλληλότητας και δοκιμή της
4. Προετοιμασία για την επόμενη επανάληψη με βάση την ανατροφοδότηση από την παραδοθείσα επαύξηση.

Το σπειροειδές μοντέλο μπορεί να διαχειριστεί την αβεβαιότητα εξαιρετικά καλά. Έτσι, είναι ιδανικό για έργα λογισμικού με ασαφή απαιτήσεις και για έργα που βρίσκονται στην έρευνα και ανάπτυξη.

### Ευέλικτο Μοντέλο

Στη σύγχρονη πραγματικότητα, η εργασία στην παραγωγή λογισμικού έχει ταχύς ρυθμούς και είναι υποκείμενο σε συνεχή ροή αλλαγών (χαρακτηριστικών, λει-





Σχήμα 2.9: Σπειροειδές Μοντέλο

τουργιών και πληροφοριακού περιεχόμενου). Έτσι, το ευέλικτο μοντέλο αποτελεί μία λογική εναλλακτική στις παραδοσιακές μεθόδους της τεχνολογίας λογισμικού.

Το μοντέλο ενθαρρύνει συνεχόμενες επαναλήψεις από ανάπτυξη και δοκιμή των λειτουργιών. Σε κάθε επανάληψη παράγεται μία επαύξηση του λογισμικού αποτελούμενη από ένα μικρό σύνολο από λειτουργίες πλήρως ολοκληρωμένες. Επίσης, η κάθε επανάληψη είναι σχεδιασμένη ώστε να είναι μικρή, διαχειρίσιμη και ολοκληρώσιμη σε λίγες εβδομάδες. Συμπεριλαμβάνει των πελάτη στην διαδικασία της ανάπτυξης και ελαχιστοποιεί τα έγγραφα χρησιμοποιώντας ανεπίσημη επικοινωνία.

Το ευέλικτο μοντέλο αν και αποτελεί ρεαλιστική προσέγγιση στην ανάπτυξη λογισμικού, βρίσκεται σε μειονεκτική θέση όταν το λογισμικό είναι σύνθετο. Επίσης, αδυνατεί να διαχειριστεί τις μεταβιβάσεις λόγω των λίγων εγγράφων, αλλά από την άλλη μπορεί να διαχειριστεί τις μεταβαλλόμενες απαιτήσεις.

Οι πιο συνηθισμένες ευέλικτες μεθοδολογίες είναι:

- **Scrum:** Αποτελείται από επαναλήψεις που λέγονται sprints. Κάθε sprint διαρκεί 2 με 4 εβδομάδες και πριν την έναρξη του γίνεται προγραμματισμός. Αφού οριστούν οι δραστηριότητες και οι εργασίες για το sprint δεν μεταβάλλονται κατά την διάρκειά του.



- **Extreme Programming (XP):** Σε αυτό το μοντέλο η επανάληψη διαρκεί 1 με 2 εβδομάδες. Χαρακτηριστικά αυτού του μοντέλου είναι ο προγραμματισμός σε ζεύγη, η ανάπτυξη οδηγούμενη από έλεγχο (test-driven development), ο αυτοματοποιημένος έλεγχος, η απλή σχεδίαση λογισμικού και τέλος οι μικρές εκδόσεις με συνεχόμενη ενσωμάτωση στο σύστημα.
- **Kanban:** Το μοντέλο Kanban επικεντρώνεται στην οπτικοποίηση, και αν υπάρχουν επαναλήψεις είναι πολύ μικρές. Στο πίνακα Kanban παριστάνονται όλες οι δραστηριότητες και εργασίες του έργου, συνοδευόμενες από τα υπεύθυνα άτομα για την διεκπεραίωσή τους και την πρόοδό τους.

#### 2.4.4 Σχεδιαστικά Πρότυπα

Πράγματι, οι σύγχρονες επιχειρήσεις στρέφονται ολοένα και περισσότερο προς τα ευέλικτα μοντέλα ανάπτυξης λογισμικού για την διαχείριση των μεταβαλλόμενων απαιτήσεων. Εντούτοις, τα ευέλικτα μοντέλα απαιτούν απλό σχεδιασμό του λογισμικού έτσι ώστε να μπορούν να επεκταθούν. Προς επίτευξη αυτού του σκοπού, καθίσταται επιτακτική η ανάγκη αναζήτησης και εφαρμογής σχεδιαστικών προτύπων.

Ο σκοπός των σχεδιαστικών προτύπων είναι να παρέχουν γενικές λύσεις σε συνηθισμένα προβλήματα του σχεδιασμού λογισμικού. Τυποποιώντας επίσημα τις λύσεις και τις σχέσεις μεταξύ τους καθίσταται

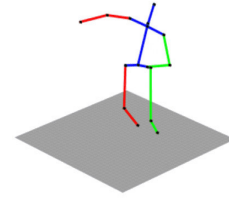
## 2.5 ΕΚΤΙΜΗΣΗ ΠΟΖΑΣ

---

Η εκτίμηση πόζας είναι μία διεργασία της υπολογιστικής όρασης όπου εκτιμάται η πόζα ενός αντικειμένου ή ανθρώπου σε μία εικόνα ή βίντεο. Το πρόβλημα της εκτίμησης πόζας περιλαμβάνει, επίσης, τον καθορισμό της θέσης και του προσανατολισμού της κάμερας σε σχέση με το αντικείμενο ή τον άνθρωπο.

Συνήθως αυτό γίνεται με την αναγνώριση, εκτίμηση θέσης και παρακολούθησης ενός αριθμού σημείων κλειδιών του αντικειμένου ή του ανθρώπου. Για τα αντικείμενα, αυτά μπορεί να είναι γωνίες ή άλλα σημαντικά χαρακτηριστικά ενώ για τον άνθρωπο τα σημεία κλειδιά αναπαριστούν κύριες αρθρώσεις όπως οι αγκώνες ή τα γόνατα.

Χαρακτηριστική είναι η διάκριση μεταξύ της 2D και της 3D εκτίμησης πόζας. Στην διδιάστατη εκτίμηση πόζας εκτιμούνται οι θέσεις των σημείων στο 2D χώρο σε σχέση με το πλαίσιο της εικόνας ή του βίντεο. Αντιθέτως, η 3D εκτίμηση πόζας προβλέπει τις συντεταγμένες των σημείων κλειδιών σε ένα τρισδιάστατο σύστημα συντεταγμένων όπως φαίνεται στο Σχήμα [2.10](#).



(α') 2D σημεία με συντεταγμένες  $(x_i, y_i)$       (β') 3D σημεία με συντεταγμένες  $(x_i, y_i, z_i)$

Σχήμα 2.10: Παράδειγμα εκτίμησης πόζας

Επιπλέον, η εκτίμηση πόζας διαφοροποιείται ως προς την αναγνώριση ενός ή περισσότερων αντικειμένων. Οι δύο προσεγγίσεις αποκαλούνται μεμονωμένη εκτίμηση πόζας, όπου αναγνωρίζεται και παρακολουθείται μόνο ένα αντικείμενο ή άνθρωπος, και πολλαπλή εκτίμηση πόζας, όπου αναγνωρίζονται και παρακολουθούνται πολλαπλά.

### 2.5.1 Μεθοδολογίες εκτίμησης πόζας

Το πρόβλημα της εκτίμησης πόζας μπορεί να λυθεί με ποικίλους τρόπους αναλόγως με την διαρρύθμιση των αισθητήρων εικόνων και την επιλογή της μεθοδολογίας. Διακρίνονται τρεις κλάσεις μεθοδολογιών [? ]:

- **Αναλυτικές ή γεωμετρικές μέθοδοι:** Απαιτούν την ρύθμιση της κάμερας ώστε η αντιστοίχιση των τρισδιάστατων σημείων της σκηνής και των δισδιάστατων σημείων της εικόνας να είναι γνωστή. Τοιουτοτρόπως, γνωρίζοντας την γεωμετρία του αντικειμένου ή προβολή του στην εικόνα της κάμερας είναι μια γνωστή συνάρτηση της πόζας του αντικειμένου. Έτσι, το πρόβλημα της εκτίμησης πόζας λύνεται αναγνωρίζοντας τα σημεία κλειδιά και στην συνέχεια λύνοντας το σύνολο των εξισώσεων που αντιστοιχίζουν τις τρισδιάστατες συντεταγμένες των σημείων με τις δισδιάστατες συντεταγμένες της εικόνας.
- **Γενετικοί αλγόριθμοι:** Στην περίπτωση που η πόζα του αντικειμένου ή του ανθρώπου δεν απαιτεί τον υπολογισμό της σε πραγματικό χρόνο μπορεί να χρησιμοποιηθεί ένας γενετικός αλγόριθμος. Η πόζα περιγράφεται από μεταβλητές, όπως για παράδειγμα οι περιστροφές των αρθρώσεων από μία στάση αναφοράς, οι οποίες χρησιμοποιούνται ως παράμετροι εισόδου του γενετικού, ορίζοντας έτσι τη συνάρτηση καταλληλότητας (fitness function) ως το σφάλμα της προβολής των εκτιμώμενων τρισδιάστατων σημείων κλειδιών από τις πραγματικές συντεταγμένες τους στο δισδιάστατο πλαίσιο της εικόνας.
- **Μηχανική μάθηση:** Οι μέθοδοι μηχανικής μάθησης χρησιμοποιούν ένα σύστημα τεχνητής νοημοσύνης το οποίο μαθαίνει την αντιστοίχιση των 2D χαρακτηριστικών της εικόνας με τις παραμέτρους μοντελοποίησης που περιγράφουν με σαφήνεια την πόζα. Εν συντομία, ένα επαρκώς μεγάλο σύνολο

από φωτογραφίες ή βίντεο αντικειμένων ή ανθρώπων σε διαφορετικές πόζες χρησιμοποιείται ως είσοδος στο σύστημα κατά την διάρκεια της φάσης εκμάθησης. Με το πέρας της εκμάθησης, το σύστημα μπορεί να εκτιμήσει την πόζα του αντικειμένου ή ανθρώπου δεδομένης της 2D εικόνας του.

Στη συγκεκριμένη εφαρμογή, απαιτείται η εκτίμηση της ανθρώπινης πόζας στον τρισδιάστατο χώρο σε πραγματικό χρόνο. Το πρόβλημα αυτό κρίνεται άκρως περίπλοκο, με την δυσκολία του να έγκειται στο γεγονός ότι η εκτίμηση της τρισδιάστατης πόζας από μία δισδιάστατη εικόνα είναι εξ ορισμού κακώς ορισμένο πρόβλημα. Αυτό γίνεται εμφανές αν σκεφτούμε ότι μία δισδιάστατη πόζα μπορεί να προκύψει από πολυάριθμες τρισδιάστατες πόζες. Ταυτόχρονα, υπάρχουν πολλές επιπλέον προκλήσεις όπως η αβεβαιότητα του περιβάλλοντος, του φόντου, του φωτισμού, η κίνηση της κάμερας, οι γρήγορες κινήσεις, οι μεταβολές των ρούχων και των σκιών που μπορεί να κάνουν την μορφή και το σχήμα του ανθρώπου να αλλάζει δραματικά με την πάροδο του χρόνου. Επιπλέον, οι ανθρώπινες αρθρώσεις είναι μικρές, οριακά εμφανής και έχουν πολλούς βαθμούς ελευθερίας δυσχεραίνοντας ακόμα περισσότερο το πρόβλημα της εκτίμησης πόζας. Ως εκ τούτου, η καταλληλότερη μεθοδολογία για την επίλυση του προβλήματος είναι η *Βαθιά Μηχανική Μάθηση*, ικανοποιώντας τόσο την απαίτηση της σχετικά ακριβής εκτίμησης των σημείων κλειδιών αλλά και διεξάγοντας την εκτίμηση σε πραγματικό χρόνο. Στη συνέχεια του κεφαλαίου λοιπόν, θα αναλύσουμε το θεωρητικό υπόβαθρο που συναντάει κανείς στις διάφορες προσεγγίσεις επίλυσης του προβλήματος εκτίμησης ανθρώπινης πόζας.

### 2.5.2 Μοντελοποίηση ανθρώπινου σώματος

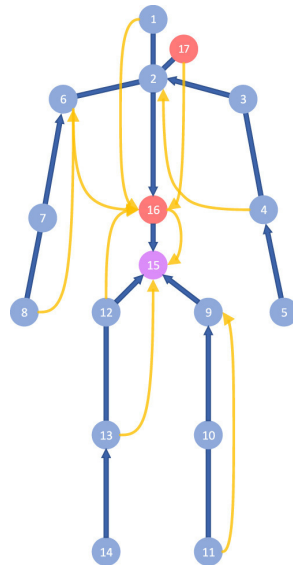
Όπως αναφέρθηκε, το ανθρώπινο σώμα είναι περίπλοκο με πολλαπλές αρθρώσεις πολλών βαθμών ελευθερίας και μεγάλου εύρους κίνησης. Τοιουτοτρόπως, κρίνεται αναγκαία η σαφής μοντελοποίηση της υποβόσκουσας δομής του, μειώνοντας αφενός την εγγενή αβεβαιότητα του προβλήματος εκτίμησης πόζας και αφετέρου αποτελώντας το μέσο για την περιγραφή της τελικής πόζας.

Οι διαφορετικές μέθοδοι εκτίμησης πόζας χρησιμοποιούν διαφορετικές μοντελοποιήσεις του ανθρώπινου σώματος ανάλογα με τις ανάγκες της εκάστοτε μεθόδου. Εντούτοις, οι κατά κόρον χρησιμοποιούμενες μοντελοποιήσεις είναι τα σκελετικά μοντέλα (*skeleton models*) και τα μοντέλα σχήματος (*shape models*). Ταυτόχρονα, αξίζει να σημειωθεί μία καινούργια μορφή αναπαράστασης του ανθρώπινου σώματος που βασίζεται στην αναπαράσταση των σημείων της επιφάνειας του σώματος. [?] ]

#### Μοντέλο σκελετού

Το Μοντέλο σκελετού είναι μια δενδρική δομή αποτελούμενη από σημεία κλειδιά του ανθρώπινου σώματος, συνδέοντας τις φυσικές παρακαείμενες αρθρώσεις, τα σημεία κλειδιά, με ακμές (για παράδειγμα η ακμή του βραχίονα συνδέει τις

αρθρώσεις-σημεία κλειδιά του καρπού και του αγκώνα). Ανάλογα με την μοντελοποίηση, ορίζονται οι γονείς πρώτης τάξης του κάθε σημείου κλειδιού ή ενδεχομένως και οι γονείς δεύτερης τάξης, όπως φαίνεται στο Σχήμα 2.11, ανάλογα με τις ανάγκες του χρησιμοποιούμενου μοντέλου μηχανικής μάθησης.



Σχήμα 2.11: Μοντέλο Σκελετού ανθρώπινου σώματος. Το σημείο κλειδί αναφοράς είναι το 15. Οι μπλε ακμές δείχνουν τους γονείς πρώτης τάξης του κάθε σημείου κλειδιού, ενώ οι κίτρινες ακμές τους γονείς δεύτερης τάξης (παραλείπονται κάποιοι για ευκρίνεια της εικόνας).

## Μοντέλο σχήματος

Το μοντέλο σχήματος αναπαριστά το ανθρώπινο σώμα ως ένα τριγωνικό πλέγμα καθορίζοντας έτσι πλήρως το σχήμα του σε οποιαδήποτε πόζα. Η μοντελοποίηση που χρησιμοποιείται στην πλειοψηφία των άρθρων εκτίμησης πόζας των τελευταίων χρόνων είναι το *skinned multi-person linear model* (SMPL) [? ]. Το μοντέλο SMPL, το οποίο φαίνεται στο Σχήμα 2.12, περιγράφει το ανθρώπινο δέρμα με 6890 κορυφές που σχηματίζουν το τριγωνικό πλέγμα. Οι κορυφές παραμετροποιούνται από τις παραμέτρους σχήματος  $\vec{\beta}$  που καθορίζουν τις αναλογίες του σώματος, όπως το ύψος και τα κιλά, και τις παραμέτρους πόζας  $\vec{\theta}$  που περιγράφουν τις περιστροφές των αρθρώσεων από μία πόζα αναφοράς.

## Επιφανειακό μοντέλο

Μία ακόμα πιο πρόσφατη μοντελοποίηση του ανθρώπινου σώματος για το πρόβλημα της εκτίμησης πόζας, αποτελεί το επιφανειακό μοντέλο. Αναπτύχθηκε με την εικασία ότι μία αραιή συσχέτιση της εικόνας με τα σημεία κλειδιά του ανθρώπινου



Σχήμα 2.12: Μοντελοποίηση SMPL. Το ανθρώπινο σώμα περιγράφεται από τις παραμέτρους σχήματος  $\vec{\beta}$  και πόζας  $\vec{\theta}$ .

σώματος δεν είναι αρκετή για να αντικατοπτρίσει τη πλήρη κατάσταση του. Τοιούτοτρόπως, στο άρθρο [? ], προτείνεται η επιφανειακή μοντελοποίηση, αποκαλούμενη DensePose, όπου εγκαθιδρύεται μία πυκνή συσχέτιση των πίξελ της εικόνας με μία επιφανειακή αντιπροσώπευση του ανθρώπινου σώματος, όπως φαίνεται στο Σχήμα 2.13

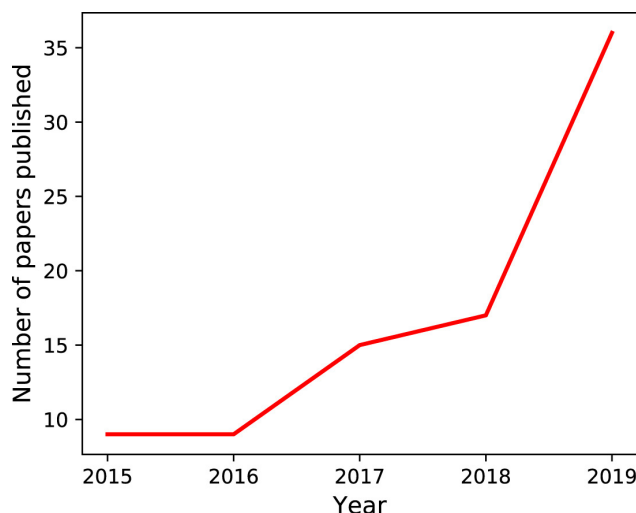


Σχήμα 2.13: Μοντελοποίηση DensePose. Στα αριστερά φαίνεται η επιφανειακή τρισιδιάστατη αναπαράσταση του ανθρώπινου σώματος. Αντιστοιχίζοντας τα σημεία της εικόνας δεξιά στην αριστερή επιφάνεια γίνεται δυνατή η εκτίμηση της πόζας.

### 2.5.3 Κατηγοριοποιήσεις λύσεων του προβλήματος εκτίμησης πόζας

Για την επίλυση του προβλήματος εκτίμησης πόζας τα τελευταία χρόνια παρουσιάζονται ολοένα και περισσότερα άρθρα όπως φαίνεται στο Σχήμα 2.14. Προφανώς, η κάθε προσέγγιση επίλυσης του προβλήματος έχει μοναδικά χαρακτηριστικά, ικανοποιώντας διαφορετικές απαιτήσεις, αλλά υπάρχουν κοινές κατευθυντήριες γραμμές πάνω στις οποίες κινούνται οι ερευνητές. Με αυτόν τον τρόπο, γίνεται

δυνατή η κατηγοριοποίηση των λύσεων του προβλήματος ως προς τα διάφορα χαρακτηριστικά τους.



Σχήμα 2.14: Αριθμός άρθρων εκτίμησης ανθρώπινης πόζας (κατακόρυφος άξονας) ανά χρόνο (οριζόντιος άξονας)

Η γενικότερη κατηγοριοποίηση αφορά την μορφή της εισόδου του μοντέλου μηχανικής μάθησης. Με άλλα λόγια, το μοντέλο μπορεί να χρησιμοποιεί για είσοδο μόνο μία εικόνα ή πολλαπλές εικόνες από διαφορετικές οπτικές (της ίδιας χρονικής στιγμής). Αντίθετα, για την αξιοποίηση της χρονικής πληροφορίας, μπορεί να χρησιμοποιηθεί ως είσοδος αλληλουχίες εικόνων, είτε μεμονωμένων είτε από διαφορετικές οπτικές. Προφανώς, οι πιο περίπλοκες εισοδοί απαιτούν περισσότερους πόρους τόσο για την ανάκτηση των εικόνων όσο και για την επεξεργασία τους, μειώνοντας ωστόσο τις ασάφειες του προβλήματος.

Επιπλέον, οι μεθοδολογίες της εκτίμησης πόζας μπορούν να διαχωριστούν ανάλογα με την ικανότητα τους στην πρόβλεψη πόζας ενός ή περισσότερων ανθρώπων. Στην πρώτη περίπτωση, ορίζεται η πρόβλεψη της πόζας μόνο για έναν άνθρωπο μέσα στην εικόνα και είναι σαφώς η απλούστερη. Στην περίπτωση εκτίμησης πόζας περισσότερων ανθρώπων το πρόβλημα δυσχεραίνεται λόγω του πολύ μεγαλύτερου χώρου καταστάσεων, της απόκρυψης ή και σύγχυσης των σημείων κλειδιών και της αναγκαιότητας διαφοροποίησης τους.

Τοιουτοτρόπως, συγκεκριμένα για το πρόβλημα εκτίμησης πόζας πολλαπλών ανθρώπων, διακρίνουμε την κατηγοριοποίηση με βάση τον τρόπο ομαδοποίησης των σημείων κλειδιών στους ανθρώπους στους οποίους ανήκουν. Οι δύο κύριες προσεγγίσεις είναι η από κάτω προς τα πάνω και η από πάνω προς τα κάτω.

- *Από κάτω προς τα πάνω:* Το μοντέλο ανιχνεύει κάθε εμφάνιση ενός συγκεκριμένου σημείου κλειδιού (όπως όλα τα αριστερά χέρια) σε μία εικόνα και στην συνέχεια προσπαθεί να ομαδοποιήσει τα σημεία κλειδιά με βάση τους ανθρώπους στους οποίους ανήκουν.
- *Από πάνω προς τα κάτω:* Αντίθετα, σε αυτή την μέθοδο, το μοντέλο αρχικά χρησιμοποιεί έναν ανιχνευτή αντικειμένων, καθορίζοντας έτσι ένα πλαί-



σιο οριοθέτησης γύρω από κάθε άνθρωπο, και έπειτα εκτιμά την θέση των σημείων κλειδιών του κάθε ανθρώπου εσωτερικά σε κάθε περικομμένη περιοχή.

Μία άλλη πιθανή κατηγοριοποίηση σχετίζεται με τα χρησιμοποιούμενα στάδια προβλέψεων έως ότου προκύψει η τελική εκτίμηση πόζας. Αναλυτικότερα, θα αναλύσουμε τις μεθόδους από τρεις κατηγορίες: εκτίμηση τρισδιάστατης πόζας απευθείας από εικόνες, ανύψωση από δισδιάστατες σε τρισδιάστατες προβλέψεις και μεθόδους βασισμένες στο μοντέλο SMPL.

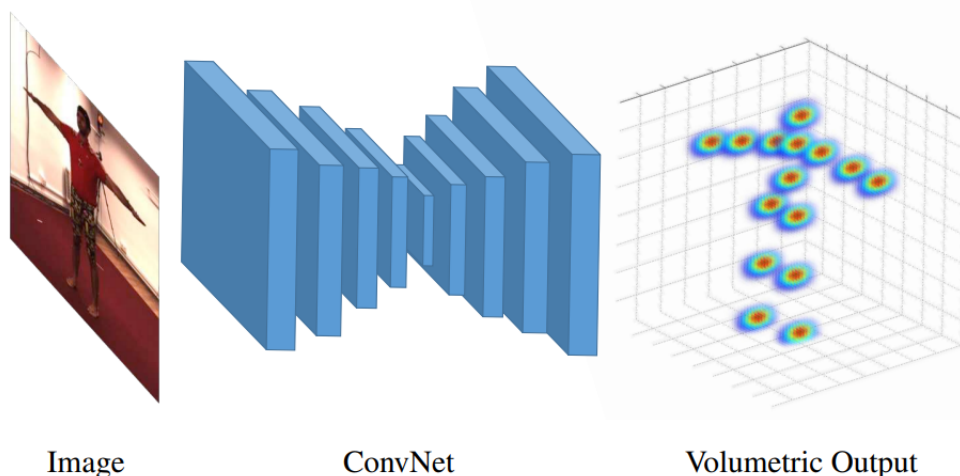
### Εκτίμηση πόζας απευθείας από εικόνες

Η πιο άμεση μεθοδολογία για την εκτίμηση της τρισδιάστατης πόζας είναι ο σχεδιασμός συνεχόμενων δικτύων για την πρόβλεψη των 3D συντεταγμένων των σημείων κλειδιών ή αρθρώσεων. Οι μέθοδοι της απευθείας εκτίμησης από εικόνες μπορούν να διαχωριστούν περαιτέρω σε δύο κλάσεις: μέθοδοι βασισμένες στον εντοπισμό και μέθοδοι βασισμένες στην παλινδρόμηση. Αξίζει να σημειωθεί ότι έχουν γίνει προσπάθειες για ενοποίηση των δύο αυτών προσεγγίσεων.

Οι μέθοδοι βασισμένες στον εντοπισμό προβλέπουν έναν πιθανοτικό θερμικό χάρτη για κάθε σημείο κλειδί και παίρνοντας την μέγιστη πιθανότητα του χάρτη καθορίζουν την θέση του σημείου κλειδιού. Για παράδειγμα στο [?] αναπαριστάται ο τρισδιάστατος χώρος σε έναν όγκο και εκπαιδεύεται ένα μοντέλο Συνελικτικού Νευρωνικού Δικτύου (*Convolutional Neural Network*, *CNN* στο καθεξής) ώστε να προβλέπονται οι ογκομετρικοί πιθανοτικοί θερμικοί χάρτες για κάθε σημείο κλειδί, όπως φαίνεται στο Σχήμα 2.15. Παρόμοιες μέθοδοι βασίζονται σε επιπλέον βήματα για την μετατροπή των θερμικών χαρτών στην τελική εκτίμηση των θέσεων των σημείων κλειδιών, συνήθως παίρνοντας την μέγιστη πιθανότητα του χάρτη, δηλαδή εφαρμόζοντας την συνάρτηση  $\text{argmax}$  (βλ. στο παράρτημα τον ορισμό ??). Βέβαια, η διαδικασία αυτή δεν είναι διαφορίσιμη, αποτελώντας τροχοπέδη στον μηχανισμό εκμάθησης των νευρωνικών δικτύων. Ταυτόχρονα, η ακρίβεια των προβλεπόμενων σημείων κλειδιών είναι ανάλογη της διακριτότητας των θερμικών χαρτών ο οποίοι έχουν εγγενή αδυναμία στη χωρική γενίκευση. Για την αύξηση της ακρίβειας των προβλέψεων, οι παραγόμενοι θερμικοί χάρτες απαιτούν αύξηση της χωρικής διακριτότητας, αυξάνοντας όμως τετραγωνικά τις απαιτήσεις σε υπολογιστική ισχύ και κατανάλωση μνήμης.

Από την άλλη μεριά, το πρόβλημα εκτίμησης πόζας μπορεί να θεωρηθεί διαισθητικά ως ένα πρόβλημα παλινδρόμησης, όπου εκτιμούνται οι συντεταγμένες θέσεων των σημείων κλειδιών ως προς την θέση ενός σημείου κλειδιού αναφοράς<sup>2</sup>. Σε αυτή την προσέγγιση, γίνεται εμφανής η καταλληλότητα επιλογής των μοντέλων σκελετού ή παρεμφερή μοντελοποιήσεις για την αναπαράσταση του ανθρώπινου σώματος. Για παράδειγμα, για να ενσωματώσουν προηγούμενη γνώση για την δομή

<sup>2</sup>Στο μοντέλο σκελετού *SMPL*, για παράδειγμα, το σημείο αναφοράς, σημείο με δείκτη (*index*) 0, είναι η λεκάνη



Σχήμα 2.15: Μέθοδος εκτίμησης πόζας βασισμένη στον εντοπισμό. Η εικόνα ως είσοδος περνάει από την αρχιτεκτονική του CNN και παράγει πιθανοτικούς χάρτες για κάθε σημείο κλειδί στον 3D χώρο.

του ανθρώπινου σώματος, στο [? ], εισάγουν ένα κινηματικό μοντέλο σκελετού, αποτελούμενο από αρθρώσεις και κόκαλα, τα οποία κόκαλα έχουν σταθερό μήκος και μπορούν να περιστραφούν γύρω από τις αρθρώσεις. Εντούτοις, το σταθερό μήκος των κοκάλων δεν ανταποκρίνεται επαρκώς στη μεταβλητότητα του ανθρώπινου σκελετού, γεγονός που υποβαθμίζει την ικανότητα γενικοποίησης του μοντέλου. Από την άλλη μεριά, στο [? ] υποθέτουν ότι στην εκτίμηση πόζας είναι πιο λογική η εκτίμηση της θέσεως των κοκάλων αντί για τις αρθρώσεις διότι η αναπαράσταση των κοκάλων καθιστά ευκολότερη την εκμάθηση του μοντέλου βαθιάς μάθησης και αντικατοπτρίζουν καλύτερα τους γεωμετρικούς περιορισμούς του ανθρώπινου σκελετού. Φυσικά, αυτή η προσέγγιση απαιτεί την μετατροπή των δεδομένων εκπαίδευσης στην σχετική μορφή αναπαράστασης τις θέσεις των κοκάλων.

Εν κατακλείδι, οι μέθοδοι εντοπισμού με θερμικούς χάρτες, αν και πιο αποδοτικοί από τις μεθόδους παλινδρόμησης, έχουν κάποια μειονεκτήματα. Η διαδικασία επιλογής της μέγιστης πιθανότητας δεν είναι διαφορίσιμη και δεν επιτρέπει την από άκρη σε άκρη εκμάθηση του μοντέλου, σε αντίθεση με τις μεθόδους παλινδρόμησης όπου αυτή η εκμάθηση είναι εφικτή.

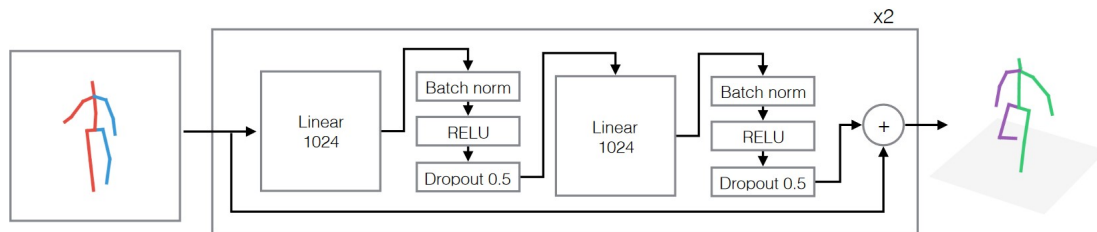
Προς αντιμετώπιση των προαναφερθέντων προβλημάτων έχουν γίνει προσπάθειες συνδυασμού των δύο παραπάνω μεθόδων. Για παράδειγμα, στο άρθρο [? ], προτείνεται η χρήση της συνάρτησης  $\text{soft-argmax}$  (αναλυτικότερα βλ. ορισμό ??) για την μετατροπή των χαρτών χαρακτηριστικών (feature maps), ή των θερμικών χαρτών, σε συντεταγμένες των σημείων κλειδιών, έχοντας ως αποτέλεσμα ένα πλήρως διαφορίσιμο σύστημα. Με παρόμοια λογική με την συνάρτηση  $\text{soft-argmax}$ , στο άρθρο [? ], εισάγουν ένα καινούργιο επίπεδο, αποκαλούμενο διαφορίσιμη χωρική σε αριθμητική μετατροπή (differentiable spatial to numerical transform, DSNT), για να διατηρήσουν την από άκρη σε άκρη δυνατότητα εκμάθησης του μοντέλου και ταυτόχρονα βελτιώνοντας την ικανότητα γενικοποίησης του.



### Ανύψωση προβλέψεων από 2D σε 3D

Εμπνευσμένοι από την ραγδαία εξέλιξη των αλγορίθμων για εκτίμηση της 2D πόζας, αρκετοί ερευνητές προσπάθησαν να χρησιμοποιήσουν τα αποτελέσματα της 2D εκτίμησης πόζας για να βελτιώσουν την ικανότητα γενικοποίησης σε δεδομένα αποκτημένα υπό καθημερινές συνθήκες.

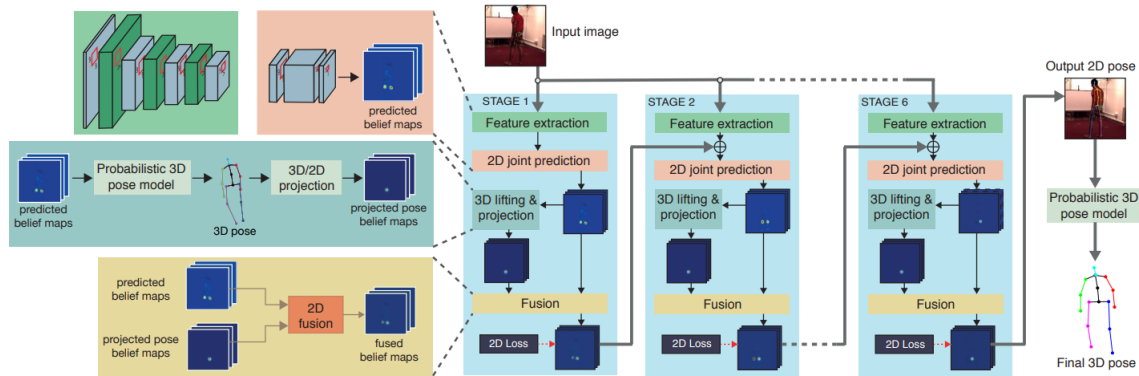
Χαρακτηριστικό παράδειγμα αποτελεί η έρευνα στο [?] όπου προτείνεται μία αρχιτεκτονική, όπως φαίνεται στο Σχήμα 2.16, επικεντρωμένη στην ανύψωση της 2D πόζας σε 3D με ένα απλό αλλά αποδοτικό νευρωνικό δίκτυο, εμφυσώντας εν-διαφέρον για περαιτέρω έρευνα στην ανύψωση της 2D πόζας στον 3D χώρο.



Σχήμα 2.16: Αρχιτεκτονική simple baseline για την εκτίμηση πόζας. Ακρογωνιαίος λίθος του νευρωνικού αποτελεί ένα γραμμικό επίπεδο, ακολουθούμενο από ένα επίπεδο κανονικοποίησης παρτίδας (batch normalization), ένα επίπεδο ενεργοποίησης με συνάρτηση ReLU (βλ. παράρτημα ??) και ένα επίπεδο εγκατάλειψης μονάδων εισόδου (dropout). Η είσοδος του συστήματος είναι ένας πίνακας με τις 2D θέσεις των σημείων κλειδιών και η έξοδος είναι οι θέσεις των σημείων κλειδιών σε 3D συντεταγμένες.

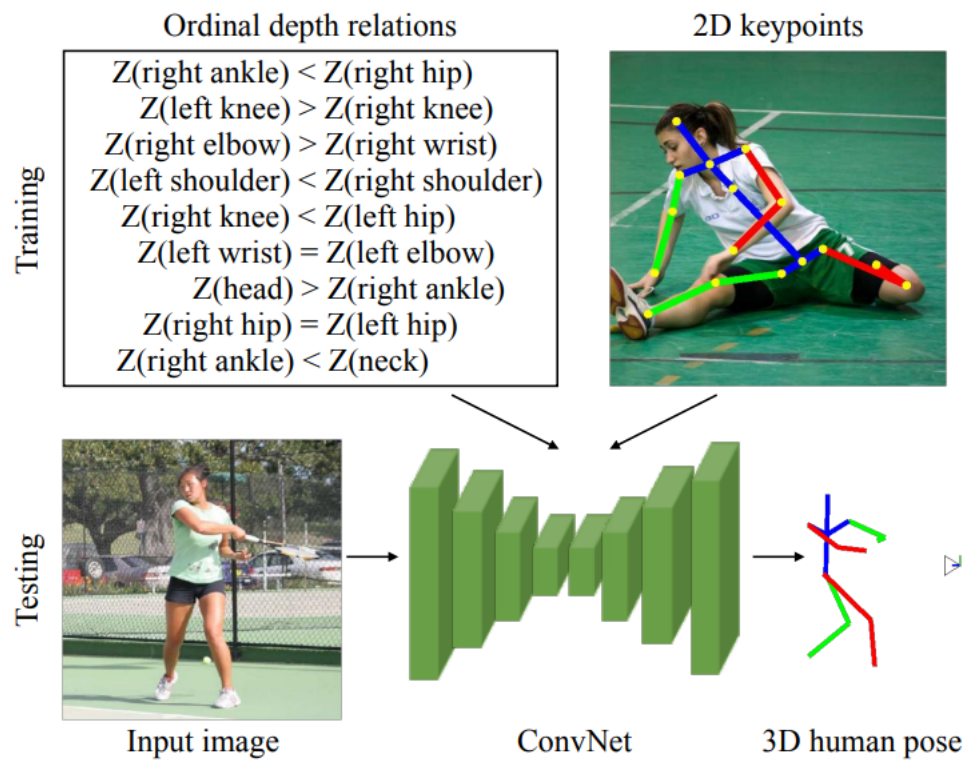
Για την αντιμετώπιση της δυσκολίας παραγωγής και σχολιασμού εξολοκλήρου 3D συνόλων δεδομένων σε διάφορες έρευνες χρησιμοποιούνται επιπλέον πληροφορίες ως ενδιάμεση επίβλεψη του συστήματος. Ένα είδος πληροφορίας που χρησιμοποιείται συχνά είναι η προβολή της προβλεπόμενης 3D πόζας στον 2D χώρο της εικόνας. Για παράδειγμα στο [?], αρχικά εκτιμάται η 2D πόζα η οποία χρησιμοποιείται για την εκτίμηση της 3D πόζας. Έπειτα, οι προβλέψεις της 3D πόζας προβάλλονται στον 2D χώρο, συνδυάζονται με τις αρχικές 2D προβλέψεις και τέλος συγκρίνονται με τους 2D σχολιασμούς του συνόλου δεδομένων (όπως φαίνεται στο Σχήμα 2.17), εκπαιδεύοντας έτσι το σύστημα. Αξίζει να σημειωθεί ότι στο στάδιο ανύψωσης των 2D προβλέψεων σε 3D ενσωματώνεται ένα επιπλέον επίπεδο στο CNN όπου αναπαριστώνται οι 3D γεωμετρικοί περιορισμοί του σώματος, ενός μοντέλου σκελετού (όπως αναπτύχθηκε στην ενότητα 2.5.2), εξασφαλίζοντας ότι οι προβλεπόμενες πόζες βρίσκονται στον χώρο των επιτρεπόμενων εκ φύσεως ποζών.

Αντίστοιχα για την αποφυγή σχολιασμού των 3D συνόλων δεδομένων, στο [?], προτείνεται μία εναλλακτική, πιο ασθενής μέθοδος επίβλεψης του συστήματος. Πιο συγκεκριμένα, για την επίβλεψη χρησιμοποιούνται οι ήδη υπάρχοντες σχολιασμοί των 2D σημείων κλειδιών σε συνδυασμό με τις σχέσεις βάθους (πιο κοντά - πιο



Σχήμα 2.17: *Lifting from the deep*: Παράδειγμα αρχιτεκτονικής ανύψωσης 2D σε 3D. Κάθε στάδιο παράγει ως έξοδο ένα σύνολο 2D πιθανοτικών χαρτών για κάθε σημείο κλειδί, οι οποίοι μαζί με την αρχική εικόνα αποτελούν την είσοδο του επόμενου σταδίου. Κάθε στάδιο μαθαίνει να συνδυάζει τους πιθανοτικούς χάρτες της 2D εκτίμησης πόζας με τους προβαλλόμενους χάρτες από την 3D εκτίμηση πόζας.

μακριά) των σημείων κλειδιών, όπως φαίνεται στο Σχήμα 2.18. Μάλιστα, παρουσιάζεται η ευελιξία αυτής της μεθόδου επίβλεψης, ενσωματώνοντάς την σε διαφορετικές διαρρυθμίσεις νευρωνικών δικτύων όπου επιτυγχάνονται ανταγωνιστικά αποτελέσματα με την επίβλεψη με πραγματικούς 3D σχολιασμούς του συνόλου δεδομένων.



Σχήμα 2.18: *Ordinal Depth Supervision: Παράδειγμα αρχιτεκτονικής ανύψωσης 2D σε 3D.* Όταν εκλείπουν οι 3D σχολιασμοί του συνόλου δεδομένου, μπορούν να χρησιμοποιηθούν οι 2D σχολιασμοί των σημείων κλειδιών σε συνδυασμό με τις σχέσεις βάθους των σημείων κλειδιών.

## Βιβλιογραφία

- [1] Boehm. “*Software Engineering*“. IEEE Transactions on Computers, C-25(12): 1226–1241, 1976.
- [2] William Stallings. “*Operating Systems: Internals and Design Principles*“. Pearson, 9 edition, 2018.
- [3] Behrouz Forouzan and Firouz Mosharraf. “*Foundations of computer science*“. Cengage Learning EMEA, 2 edition, 2008.
- [4] Noam Nisan and Shimon Schocken. “*The Elements of Computing Systems: Building a Modern Computer from First Principles*“. The MIT Press, 2 edition, 2021.
- [5] Mark Segal and Kurt Akeley. “*The OpenGL Graphics System: A Specification (Version 4.6 (Core Profile) - October 22, 2019)*“, 2021. URL <https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf>.
- [6] Abiy Biru Chebudie, Roberto Minerva, and Domenico Rotondi. “*Towards a definition of the Internet of Things (IoT)*“. PhD thesis, 08 2014.
- [7] Pallavi Sethi and Smruti R. Sarangi. “*Internet of Things: Architectures, Protocols, and Applications*“. JECE, 2017, January 2017. ISSN 2090-0147.
- [8] Hans van Vliet. “*Software Engineering: Principles and Practice*“. Wiley, 2007.
- [9] Roger S. Pressman and Bruce R. Maxim. “*Software Engineering: A practitioner’s approach*“. McGraw-Hill Education, 8 edition, 2014. ISBN 978-0-07-802212-8.