ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΑΞΙΟΠΟΙΗΣΗ ΜΕΓΑΛΩΝ ΓΛΩΣΣΙΚΩΝ ΜΟΝΤΕΛΩΝ ΓΙΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΑΥΤΟΜΑΤΩΝ ΕΛΕΓΧΩΝ ΑΠΟΔΟΧΗΣ ΛΟΓΙΣΜΙΚΟΥ»

ΠΛΑΤΙΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: P3200157

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΔΙΑΜΑΝΤΙΔΗΣ ΝΙΚΟΛΑΟΣ

ΑΘΗΝΑ, ΑΥΓΟΥΣΤΟΣ 2024

Περίληψη εργασίας

Στη σύγχρονη ανάπτυξη λογισμικού, τα μεγάλα γλωσσικά μοντέλα παίζουν κρίσιμο ρόλο στην αυτοματοποίηση της παραγωγής κώδικα και στη διαχείριση των απαιτήσεων. Αυτά τα μοντέλα, τα οποία βασίζονται σε τεχνικές τεχνητής νοημοσύνης, μπορούν να κατανοούν και να παράγουν ανθρώπινη γλώσσα με μεγάλη ακρίβεια, βελτιώνοντας την ποιότητα του λογισμικού.

Το Behavior-Driven Development (BDD) είναι μια μεθοδολογία η οποία επικεντρώνεται στη συνεργασία μεταξύ προγραμματιστών και χρηστών για την κατανόηση των απαιτήσεων μέσω δημιουργίας σεναρίων σε φυσική γλώσσα. Το Cucumber είναι ένα εργαλείο που υποστηρίζει το BDD με τη συγγραφή και αυτοματοποίηση αυτών των σεναρίων.

Η παρούσα εργασία έχει ως σκοπό την αξιολόγηση με βάση αντικειμενικών κριτηρίων των μεγάλων γλωσσικών μοντέλων στην παραγωγή αυτοματοποιημένου κώδικα μέσω BDD. Η μελέτη αξιολογεί τα μοντέλα GPT-3.5, GPT-4, GPT-4o και GitHub Copilot, εστιάζοντας στη δημιουργία Step Definitions και την κατανόηση μεταβλητών. Η μεθοδολογία περιλαμβάνει τέσσερις φάσεις πειραμάτων, όπου κάθε φάση αποτελείται από διαφορετική ποσότητα προϋπάρχουσας γνώσης και διαφορετικές τεχνικές παρουσίασης της πληροφορίας στο μοντέλο, με σκοπό την εύρεση του βέλτιστου τρόπου επικοινωνίας με αυτό.

Τα ευρήματα δείχνουν ότι το GPT-4o ξεχωρίζει στην παραγωγή Step Definitions και κατανόηση απαιτήσεων, με το GitHub Copilot να ακολουθεί σε μικρότερο βαθμό. Οι στρατηγικές της παρουσίασης των σεναρίων BDD σε ένα ενιαίο μήνυμα και της εντολής στο μοντέλα να παράγει πρώτα τον κώδικα για Domain classes/Data Access Objects/Services αποδείχθηκαν ιδιαίτερα αποτελεσματικές, ενώ η προσθήκη πληροφοριών για τις κλάσεις Domain βελτίωσε την ποιότητα των αποτελεσμάτων.

Abstract of thesis

In modern software development, large language models play a crucial role in automating code and managing requirements. These models, which are based on artificial intelligence techniques, can understand and generate human language with high accuracy, thus improving software quality.

Behavior-Driven Development (BDD) is a methodology that focuses on collaboration between developers and end users to understand requirements by creating scenarios in natural language. Cucumber is a tool that supports BDD by writing and automating these scenarios.

This thesis wishes to objectively evaluate large language models in the generation of automated code through BDD. The study assesses the GPT-3.5, GPT-4, GPT-4o, and GitHub Copilot models, focusing on the creation of Step Definitions and understanding variables. The methodology includes four phases of experiments, each involving different amounts of prior knowledge and different techniques for presenting information to the model, with the goal of finding the optimal way to communicate with it.

The findings show that GPT-4o excels in producing Step Definitions and understanding requirements, with GitHub Copilot following to a lesser extent. Strategies such as presenting BDD scenarios in a single prompt and instructing the model to first generate code for Domain classes/Data Access Objects/Services proved to be very effective, while adding information about the Domain classes improved the quality of the results.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Κατάλογος εικόνων

Κατάλογος πινάκων

1. Εισαγωγή

1.1 Μεγάλα γλωσσικά μοντέλα, BDD και αυτοματοποίηση κώδικα

Στην σύγχρονη εποχή της τεχνολογίας, η τεχνητή νοημοσύνη και τα μεγάλα γλωσσικά μοντέλα, όπως το GPT-3.5, το GPT-4 και το GPT-4o της OpenAI, έχουν γίνει πολύ σημαντικά εργαλεία σε πολλούς τομείς της καθημερινότητας και της τεχνολογίας της επιστήμης υπολογιστών. Συγκεκριμένα, τα μεγάλα γλωσσικά μοντέλα έχουν αλλάξει ριζικά τον τρόπο που οι άνθρωποι χειρίζονται την ανάπτυξη λογισμικού, αφού έχουν φέρει την επανάσταση στην επεξεργασία της φυσικής γλώσσας, προσφέροντας δυνατότητες που πριν από λίγα χρόνια θεωρούνταν αδιανόητες. Αυτά τα μοντέλα, μέσω της εκπαίδευσής τους σε τεράστιους όγκους δεδομένων, έχουν αναπτύξει ικανότητες κατανόησης της φυσικής γλώσσας σε επίπεδο που δεν υπάρχει πια διαφοροποίηση από τους ανθρώπους και την ανθρώπινη επικοινωνία.

Η προοπτική που παρουσιάζουν τα μεγάλα γλωσσικά μοντέλα αυτά στην βελτίωση της διαδικασίας της ανάπτυξης λογισμικού παρουσιάζει ιδιαίτερο ενδιαφέρον. Ειδικότερα, η χρήση αυτών των μοντέλων για την δημιουργία αυτοματοποιημένου κώδικα σε σενάρια Behavior-Driven Development έχει προκαλέσει ιδιαίτερο ενδιαφέρον, καθώς μπορεί να μειώσει δραστικά τον χρόνο ανάπτυξης και να βελτιώσει σε μεγάλο βαθμό την ποιότητα του κώδικα. Το BDD (Behavior-Driven Development) αποτελεί μία μεθοδολογία ανάπτυξης λογισμικού που εστιάζει στην κατανόηση και την επαλήθευση της συμπεριφοράς του συστήματος από την πλευρά του χρήστη, απαιτώντας σαφή και κατανοητή περιγραφή των απαιτήσεων και των λειτουργιών του συστήματος, ενώ παράλληλα προωθεί την συνεργασία μεταξύ των διαφορετικών μερών ενός συστήματος λογισμικού.

Παρά τη σημαντική πρόοδο που έχουν επιτύχει τα μεγάλα γλωσσικά μοντέλα, η εφαρμογή τους σε συγκεκριμένες μεθοδολογίες ανάπτυξης, όπως το BDD, ενδέχεται να αντιμετωπίζει προκλήσεις. Τα μοντέλα αυτά πρέπει να είναι σε θέση να κατανοούν και να παράγουν ακριβή και λειτουργικό κώδικα που ανταποκρίνεται στις απαιτήσεις των BDD, συμπεριλαμβανομένων των Step Definitions και των απαιτήσεων του BDD. Ειδικότερα, οι προκλήσεις περιλαμβάνουν τη σωστή κατανόηση των απαιτήσεων του χρήστη και τη δημιουργία κώδικα που να είναι σύμφωνος με τα σενάρια BDD που έχουν δοθεί.

1.2 Σκοπός της παρούσας διπλωματικής εργασίας

Η παρούσα διπλωματική εργασία ασχολείται με το πρόβλημα της αξιολόγησης της απόδοσης των μεγάλων γλωσσικών μοντέλων στην παραγωγή αυτοματοποιημένου κώδικα για σενάρια BDD, έχοντας όμως κάθε φορά ένα διαφορετικό μέγεθος προϋπάρχουσας γνώσης δοσμένο στο σύστημα. Το μέγεθος της προϋπάρχουσας αυτής γνώσης χωρίστηκε σε τέσσερις φάσεις, με κάθε μία να προσθέτει επιπλέον πληροφορίες από την προηγούμενη. Επίσης, η εργασία αυτή έχει και ως σκοπό την εύρεση μίας βέλτιστης μεθοδολογίας για την επικοινωνία με τα μεγάλα γλωσσικά μοντέλα και της παρουσίασης της γνώσης σε αυτά με τέτοιο τρόπο, ώστε να παράγονται τα βέλτιστα αποτελέσματα στον λιγότερο χρόνο.

1.3 Δομή

Αρχικά, παρουσιάζεται μια ανάλυση της έννοιας των μεγάλων γλωσσικών μοντέλων, περιγράφοντας τις αρχές τους, την αρχιτεκτονική τους, τον τρόπο με τον οποίο λειτουργούν αλλά και την ιστορία τους μέχρι το σήμερα. Στην συνέχεια, εξετάζεται η τεχνολογία πίσω από την λειτουργία τους, όπως η εκπαίδευσή τους και οι αλγόριθμοι που χρησιμοποιούνται πίσω από αυτά τα πολύ δημοφιλή μοντέλα.

Επιπρόσθετα, παρουσιάζεται η χρήση των μεγάλων γλωσσικών μοντέλων στον τομέα της ανάπτυξης λογισμικού, όπως η χρήση τους για την δημιουργία αυτοματοποιημένου κώδικα, την βελτίωση της ποιότητας του κώδικα και την ενίσχυση της παραγωγικότητας των προγραμματιστών. Εξετάζονται παραδείγματα εφαρμογών τους, τα πλεονεκτήματα , οι προκλήσεις αλλά και οι μελλοντικές προοπτικές τους για βελτίωση.

Συνεχίζοντας, γίνεται μία αναλυτική περιγραφή της έννοιας της μεθοδολογίας Behavior-Driven Development, εξηγώντας τον βασικό σκοπό της, ποια είναι τα χαρακτηριστικά της, οι αρχές της και πως εφαρμόζεται στον τομέα της ανάπτυξης λογισμικού. Ακόμη, παρουσιάζεται ο τρόπος χρήσης του εργαλείου Cucumber σε συνδυασμό με το BDD, ο τρόπος δημιουργίας αυτοματοποιημένων τεστ με το εργαλείο αυτό αλλά και όλες οι απαιτούμενες θεωρητικές γνώσεις που απαιτούνται για την υιοθέτηση του εργαλείου αυτού, ακολουθούμενα από πραγματικά παραδείγματα.

Τέλος, παρουσιάζεται η χρήση των μεγάλων γλωσσικών μοντέλων για την υποστήριξη του BDD.

Συγκεκριμένα, παρουσιάζεται η μεθοδολογία που ακολουθήθηκε κατά την διεξαγωγή του πειράματος και της αξιολόγησης των μεγάλων γλωσσικών μοντέλων, τα κριτήρια που χρησιμοποιήθηκαν για την αξιολόγηση αυτή αλλά και τα συμπεράσματα για τα τελικά αποτελέσματα (σύνδεσμος εργασίας στο GitHub)

2. Μεγάλα Γλωσσικά Μοντέλα

2.1 Τι είναι ένα νευρωνικό δίκτυο και πώς λειτουργεί

2.1.1 Ορισμός νευρωνικών δικτύων

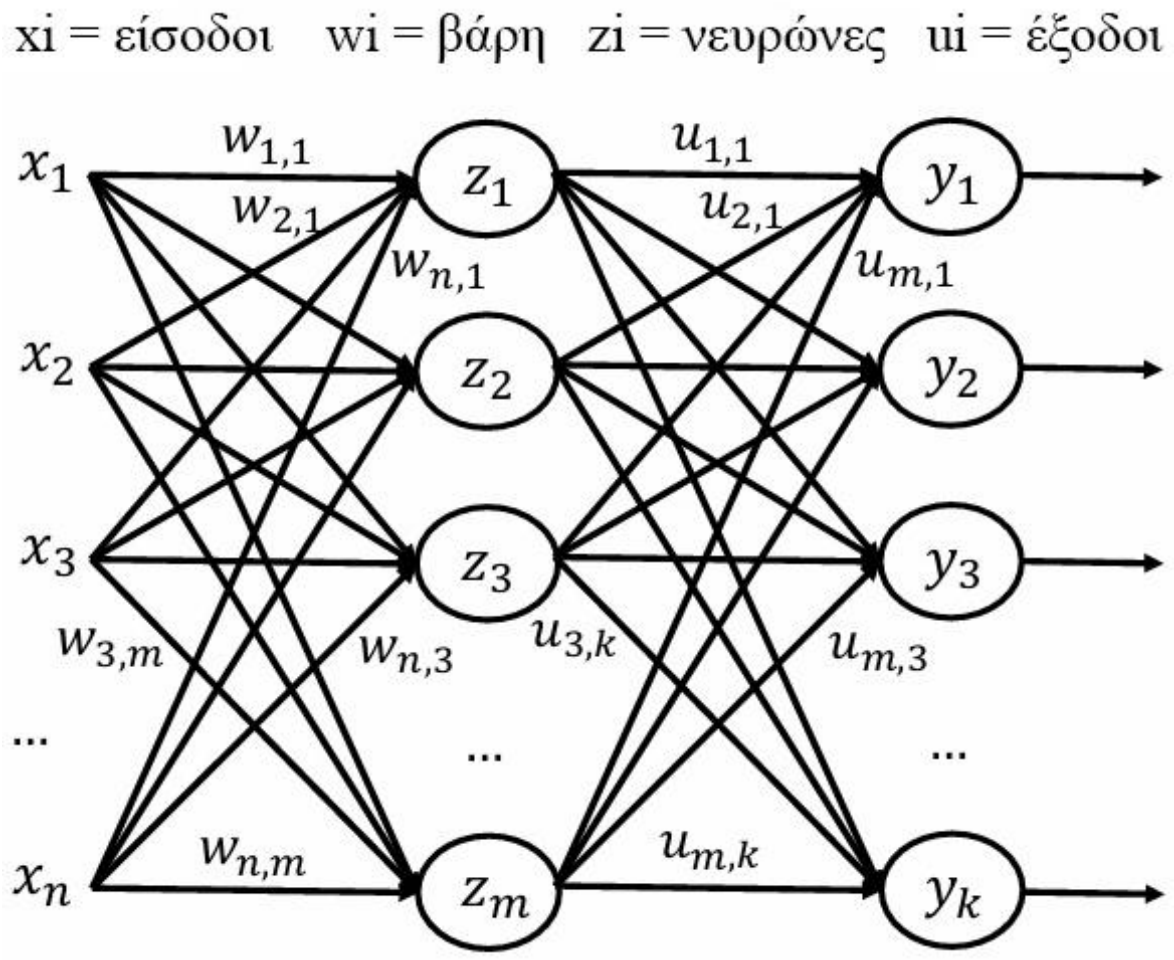
Ξεκινώντας τη συζήτηση για τα μεγάλα γλωσσικά μοντέλα και τη χρήση τους, δεν θα μπορούσε να παραληφθεί η αναφορά στη βασική συνιστώσα που τα αποτελεί, το **«νευρωνικό δίκτυο»** αλλά και στους τρόπους με τους οποίους αυτό λειτουργεί. Ένα νευρωνικό δίκτυο, όπως γίνεται αντιληπτό από το όνομά του, είναι ένα δίκτυο πολλών συνδεδεμένων νευρώνων, χωρισμένων σε διαφορετικά «στρώματα». Κάθε νευρώνας μπορεί να θεωρηθεί ως μία μονάδα που λαμβάνει κάποιες εισόδους, εκτελεί κάποιους πολύ απλούς υπολογισμούς με αυτές και στη συνέχεια παράγει κάποια έξοδο, ένα νούμερο, το οποίο με την σειρά του περνάει ως είσοδος στους επόμενους νευρώνες.

Οι απλοί αυτοί υπολογισμοί είναι το άθροισμα των εισόδων πολλαπλασιασμένοι με έναν αριθμό, το λεγόμενο «βάρος», το οποίο το νευρωνικό δίκτυο μαθαίνει από χιλιάδες δεδομένα κατά την εκπαίδευσή του. Στην συνέχεια, αφού εφαρμοστεί ένας αριθμητικός μετασχηματισμός, το αποτέλεσμα αυτό μεταφέρεται στους νευρώνες του επόμενου στρώματος ως είσοδος και η διαδικασία αυτή συνεχίζεται μέχρι το τελευταίο στρώμα νευρώνων του νευρωνικού.

Για παράδειγμα, ένας νευρώνας να προωθεί στους νευρώνες του επόμενου στρώματος τον αριθμό μηδέν εάν οι είσοδοι που δέχτηκε ήταν αρνητικοί ή κοντά στο μηδέν, και τον αριθμό ένα στην αντίθετη περίπτωση. Αυτό είναι γνωστό ως «σιγμοειδής συνάρτηση ενεργοποίησης».

Συνεχίζοντας, τα μοντέλα βαθιάς μάθησης χρησιμοποιούν εκατομμύρια νευρώνες, χωρισμένους σε πολυάριθμα στρώματα με δισεκατομμύρια βάρη και πολύπλοκες διατάξεις νευρώνων, αλλά η βασική ιδέα παραμένει η ίδια. (Ανδρουτσόπουλος, 2024). Στην **Εικόνα 1**, παρουσιάζεται μία απλοϊκή μορφή ενός νευρωνικού δικτύου, η οποία αποτελείται από ένα στρώμα εισόδων, ένα εσωτερικό στρώμα και ένα στρώμα εξόδων.

Εικόνα 1. Παράδειγμα απλού νευρωνικού δικτύου



Σημείωση: Ανάκτηση από «Τεχνητή Νοημοσύνη και Μεγάλα Γλωσσικά Μοντέλα», Ι.Ανδρουτσόπουλος, ΟΠΑ News Εφημερίδα Οικονομικού Πανεπιστημίου Αθηνών Τεύχος 51, 2024, σελ.8,(σύνδεσμος).

2.1.2 Ορισμός της μείωσης βαρών με καθοδική κλίση

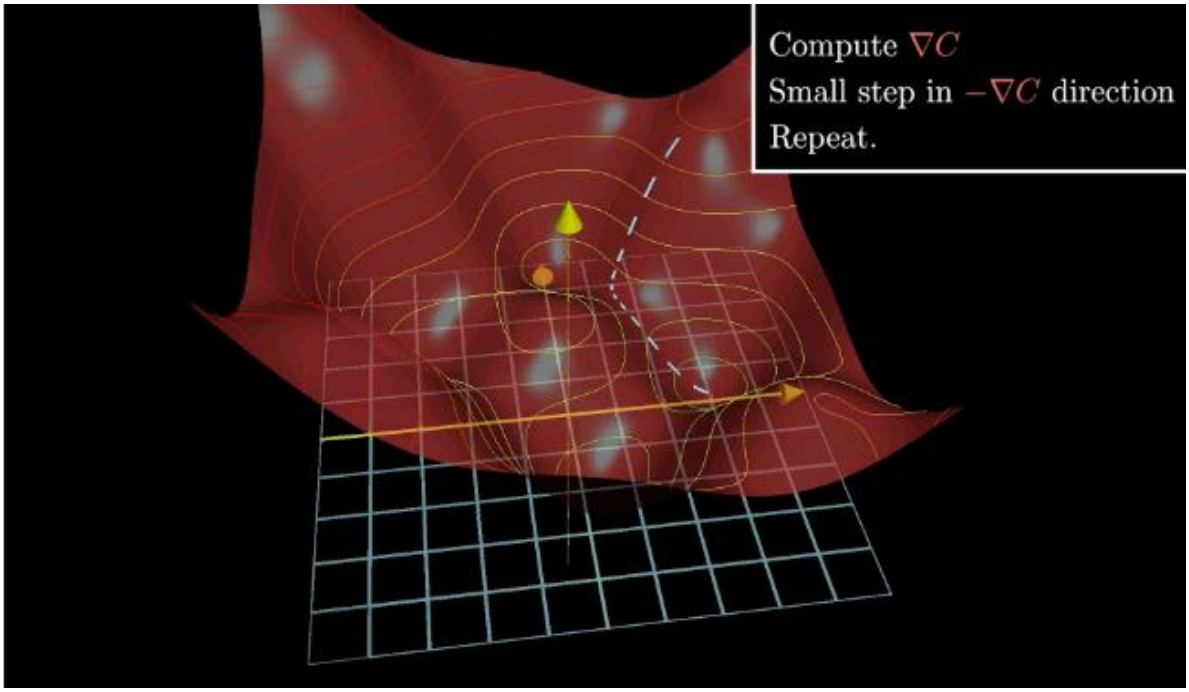
Αυτό που κάνει τα νευρωνικά δίκτυα τόσο ενδιαφέροντα, και κατά συνέπεια τη μηχανική μάθηση, είναι ότι στην ουσία, ποτέ δεν γράφεται κάποιος αλγόριθμος που να ορίζει συγκεκριμένα τι θα πρέπει να κάνει ένα νευρωνικό δίκτυο και πως θα πρέπει να παράγει αποτελέσματα. Αντίθετα, γράφεται ένας αλγόριθμος που, μέσω της εισαγωγής εκατομμυρίων παραδειγμάτων και δεδομένων με τις ορθές «ετικέτες» τους (δηλαδή το επιθυμητό αποτέλεσμα), μπορεί να

μεταβάλλει τα εκατομμύρια βάρη από τα οποία αποτελείται το δίκτυο, ώστε να αποδίδει καλύτερα στα παραδείγματα αυτά.

Τα δεδομένα αυτά ονομάζονται «δεδομένα εκπαίδευσης», και μαζί με τα «δεδομένα δοκιμής», (παραδείγματα που το νευρωνικό δίκτυο δεν έχει «ξαναδεί» και χρησιμοποιούνται για την αξιολόγηση της αποτελεσματικότητάς του), αποτελούν την συνολική αξιολόγηση του συστήματος. Το πρόβλημα αυτό της μεταβολής των βαρών που καλείται να λύσει ο αλγόριθμος καταλήγει να είναι η εύρεση του ελαχίστου μιας «συνάρτησης κόστους».

Η συνάρτηση αυτή υπολογίζεται με βάση τα αποτελέσματα που παράγει το νευρωνικό δίκτυο και τα επιθυμητά αποτελέσματα που έχουμε για κάθε παράδειγμα, και συνεπώς έχει μεγάλη τιμή εάν τα αποτελέσματα διαφέρουν σε μεγάλο βαθμό από τα επιθυμητά, και μικρή τιμή στην αντίθετη περίπτωση. Σκοπός του αλγορίθμου είναι, με βάση τον μέσο όρο όλων αυτών των τιμών κόστους για κάθε παράδειγμα, να προσπαθήσει να μεταβάλλει τις τιμές των βαρών της έτσι ώστε η συνάρτηση κόστους να φτάσει σε ένα τοπικό ελάχιστο.

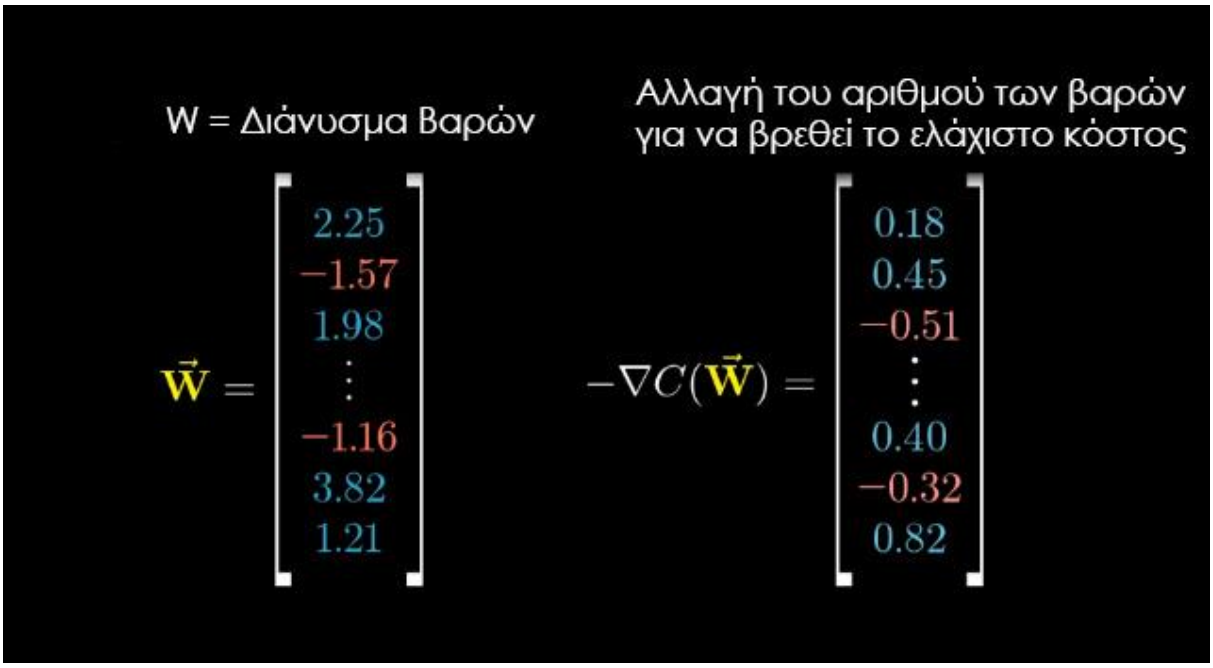
Εικόνα 2. Παράδειγμα αναπαράστασης του gradient descent



Σημείωση: Ανάκτηση από «Gradient descent, how neural networks learn», G. Sanderson-3blue1brown,2017 (σύνδεσμος).

Η τεχνική αυτή ονομάζεται «**καθοδική κλίση**» ή όπως είναι γνωστή, gradient descent, καθώς προσπαθεί να βρει ένα τοπικό ελάχιστο μίας συνάρτησης με πολλές χιλιάδες μεταβλητές (τις εισόδους και τα βάρη). Εάν αναπαρασταθεί σε έναν διανυσματικό χώρο, η συνάρτηση έχει την εικόνα ενός «γεωγραφικού τοπίου» στο οποίο πρέπει να βρεθεί ένα τοπικό «χαμηλότερο σημείο», όπως φαίνεται και στην ***Εικόνα 2***. Χρησιμοποιώντας μαθηματικές έννοιες, όλα τα παραπάνω καταλήγουν τελικά στην εύρεση του αρνητικού του «ανάδελτα» ή ∇ της συνάρτησης κόστους, το οποίο δείχνει προς την κατεύθυνση της πιο απότομης μείωσης της συνάρτησης (Sanderson, 2017).

Εικόνα 3. Παράδειγμα μαθηματικής αναπαράστασης του ανάδελτα



Σημείωση: Ανάκτηση από «Gradient descent, how neural networks learn», G. Sanderson--3blue1brown,2017 (σύνδεσμος).

Κατά συνέπεια, σε αυτό το σημείο θα αναφερθεί περιληπτικά ο ορισμός του αλγορίθμου που προσπαθεί να επιτύχει όλα τα παραπάνω, δηλαδή να υπολογίσει αυτό το τοπικό ελάχιστο της συνάρτησης κόστους, ο οποίος αναφέρεται ως «**αλγόριθμος ανάστροφης μετάδοσης**».

2.1.3 Ορισμός του αλγορίθμου ανάστροφης μετάδοσης

Όπως παρουσιάστηκε παραπάνω, ο αλγόριθμος ανάστροφης μετάδοσης είναι ένας αλγόριθμος που βρίσκει το τοπικό ελάχιστο μέσω του υπολογισμού του αρνητικού ανάδελτα μίας συνάρτησης κόστους. Αρχικά, ο αλγόριθμος αρχικοποιεί όλα τα βάρη του νευρωνικού δικτύου με τυχαίες

μικρές τιμές. Για μία δεδομένη είσοδο ή παράδειγμα εκπαίδευσης, υπολογίζει το συνολικό σφάλμα ή τη συνάρτηση κόστους στην τελική έξοδο, συγκρίνοντας την πραγματική έξοδο με την επιθυμητή έξοδο. Στην συνέχεια, το σφάλμα μεταδίδεται από την έξοδο προς την είσοδο, υπολογίζοντας παράλληλα τους παραγώγους ως προς κάθε ξεχωριστό βάρος με τον κανόνα της αλυσίδας. Κάθε βάρος ενημερώνεται ώστε να κατευθύνεται, με την χρήση της καθοδικής κλίσης, προς την μείωση του σφάλματος. Η διαδικασία αυτή επαναλαμβάνεται για κάθε παράδειγμα εκπαίδευσης που δίνεται στο νευρωνικό δίκτυο, οι οποίες ονομάζονται «εποχές». Η εκπαίδευση τελειώνει είτε όταν το σύστημα ξεπεράσει έναν μέγιστο αριθμό εποχών, είτε όταν το συνολικό σφάλμα μειωθεί σε έναν επιθυμητό αριθμό (Ανδρουτσόπουλος, 2023).

2.2 Τι είναι ένα μεγάλο γλωσσικό μοντέλο

2.2.1 Ορισμός μεγάλου γλωσσικού μοντέλου

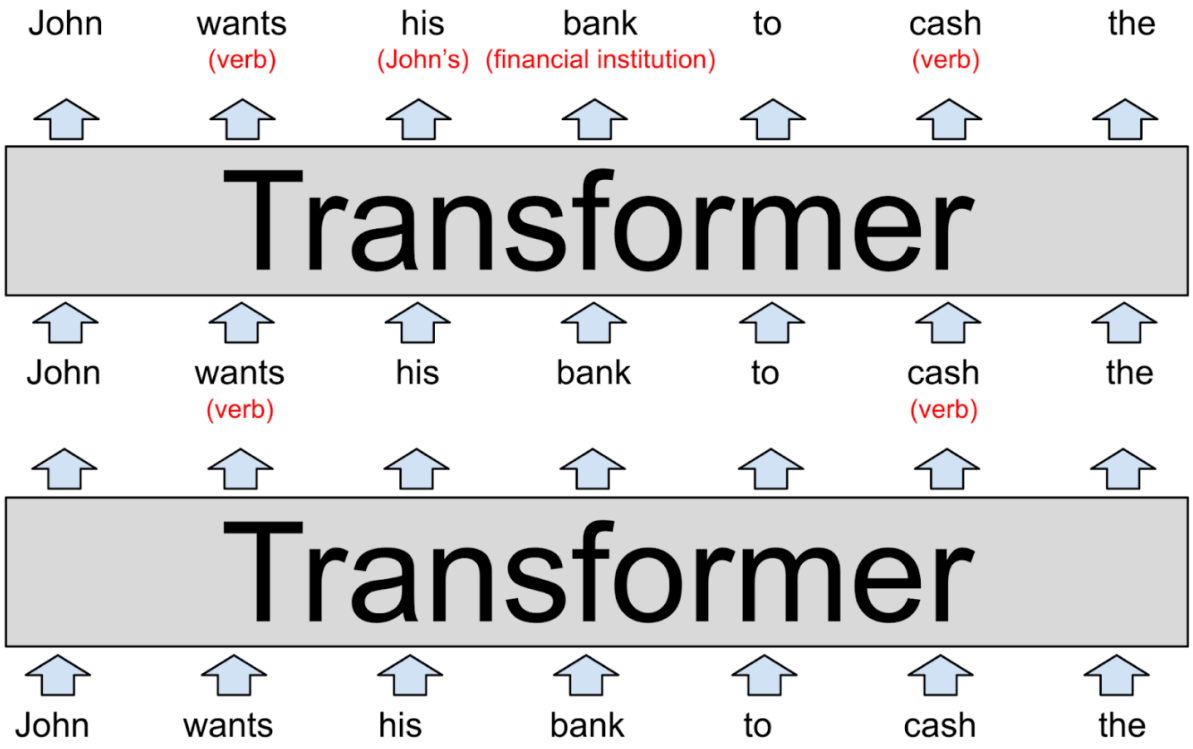
Όπως αναφέρει ο Ανδρουτσόπουλος(2024), ένα μεγάλο γλωσσικό μοντέλο είναι ένα νευρωνικό δίκτυο που δέχεται ως εισόδους λέξεις ή καλύτερα «tokens», σε μορφή αριθμών, τα οποία μπορεί να αποτελούν ένα ημιτελές κείμενο, και παράγει ως εξόδους όλες τις πιθανές λέξεις που θα μπορούσαν να είναι η επόμενη λέξη, επιλέγοντας αυτήν με την μεγαλύτερη πιθανότητα ορθότητας. Έτσι, βασιζόμενοι στο μεγάλο γλωσσικό μοντέλο, μπορούμε να πάρουμε την νέα πρόταση που παρήγαγε, να την ξαναδώσουμε σαν είσοδο και να παράγει ξανά άλλη μία επόμενη πιθανή λέξη. Μετά από έναν αριθμό επαναλήψεων της διαδικασίας αυτής, παράγεται μια ολοκληρωμένη απάντηση που βγάζει νόημα.

2.2.2 Τι κάνει ένα Transformer

Πιο συγκεκριμένα, τα μεγάλα γλωσσικά μοντέλα διασπούν το κείμενο που δέχονται ως είσοδο σε διαφορετικά tokens, τα οποία δεν είναι απαραίτητα λέξεις, αλλά μπορεί να είναι και κομμάτια λέξεων, και στην συνέχεια τα αναπαριστούν σε διανύσματα. Τα διανύσματα αυτά αποτελούνται από χιλιάδες μεταβλητές και αριθμούς και τοποθετούνται σε έναν πολυδιάστατο χώρο, όπου λέξεις με παρόμοια σημασία, όπως «γάτα» και «σκύλος», βρίσκονται πολύ κοντά. Η χρήση διανυσμάτων επιτρέπει στα μεγάλα γλωσσικά μοντέλα να πραγματοποιούν μαθηματικές πράξεις που αποκαλύπτουν σχέσεις μεταξύ λέξεων. Για παράδειγμα έχει διαπιστωθεί ότι αν από το διάνυσμα της λέξης "μεγαλύτερος" αφαιρεθεί το διάνυσμα της λέξης "μεγάλος" και προστεθεί το διάνυσμα της λέξης "μικρός", το αποτέλεσμα θα είναι το διάνυσμα της λέξης "μικρότερος".

Τα μεγάλα γλωσσικά μοντέλα μπορούν να αναπαριστούν τις λέξεις με διαφορετικά διανύσματα ανάλογα με τα συμφραζόμενα. Αυτό επιτυγχάνεται με τη χρήση ενός αρχιτεκτονικού μοντέλου νευρωνικού δικτύου, γνωστού ως «**transformer»,** που ενημερώνει τα διανύσματα των λέξεων μέσω πολλαπλών επιπέδων. Κάθε μεγάλο γλωσσικό μοντέλο αποτελείται από πολλά στρώματα transformers συνδεδεμένα μεταξύ τους, με σκοπό να εμπλουτίζουν κάθε token με πληροφορίες βάσει των συμφραζομένων, τροποποιώντας έτσι το διάνυσμα του ώστε να έχει την ορθή σημασία που αποκαλύπτεται από τα συμφραζόμενα. Κάθε transformer προσθέτει πληροφορία

Εικόνα 4. Παράδειγμα επικοινωνίας των transformers σε ένα μεγάλο γλωσσικό μοντέλο



Σημείωση: Ανάκτηση από «Large Language Models, explained with a minimum of math and jargon», T. Lee and S. Trott, 2023, (σύνδεσμος).

και βελτιώνει τα διανύσματα έως το τελικό στρώμα (Lee, 2023).

2.2.3 Τι είναι το Attention Layer

Ένα κύριο συστατικό που επιτρέπει στο transformer να εμπλουτίζει τα διανύσματα των λέξεων είναι το **«attention layer».** Συγκεκριμένα, το στοιχείο αυτό επιτρέπει στα tokens να ανταλλάσσουν πληροφορίες μεταξύ τους και να εμπλουτίζουν το ένα το άλλο, έτσι ώστε να προκύπτει η σωστή σημασία κάθε λέξης. Για παράδειγμα, η λέξη «μοντέλο» έχει διαφορετική

σημασία ανάλογα τα συμφραζόμενα, όπως στις φράσεις «μοντέλο μαθηματικών» και «μοντέλο του Χόλυγουντ». Η λειτουργία του attention layer είναι να πραγματοποιεί τον σωστό αυτό διαχωρισμό για κάθε διαφορετικό token με βάση τις λέξεις που το περιτριγυρίζουν. Η διαδικασία αυτή επιτρέπει στα μεγάλα γλωσσικά μοντέλα να μαντεύουν σωστά την επόμενη λέξη σε κάθε κείμενο (Lee, 2023)

2.2.4 Τι είναι το Feed Forward Step

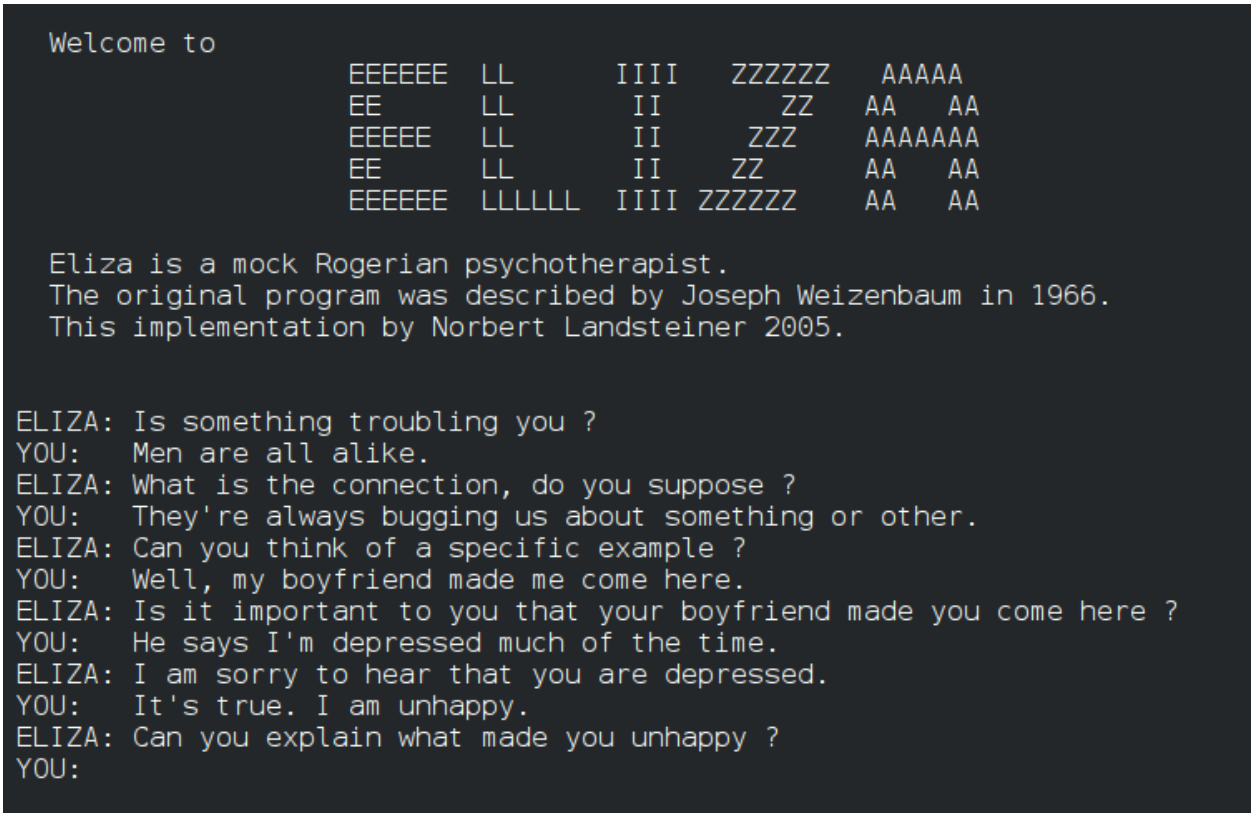
Μετά τη μεταφορά πληροφοριών ανάμεσα στα διανύσματα λέξεων από τα attention heads, το transformer περιλαμβάνει ένα ακόμα συστατικό, το στρώμα **«feed forward»**. Το στρώμα αυτό «εξετάζει» κάθε διάνυσμα λέξης και προσπαθεί να προβλέψει την επόμενη λέξη. Σε αυτό το στάδιο, δεν γίνεται ανταλλαγή πληροφοριών μεταξύ των λέξεων, το στρώμα feed forward αναλύει κάθε λέξη μεμονωμένα. Ωστόσο, το στρώμα feed forward έχει πρόσβαση σε οποιαδήποτε πληροφορία έχει αντιγραφεί από όλα τα προηγούμενα attention heads, ώστε να μπορέσει αποτελεσματικότερα να προβλέψει την επόμενη λέξη (Lee, 2023).

2.3 Ιστορία και εξέλιξη των μεγάλων γλωσσικών μοντέλων

2.3.1 Το chatbot ELIZA

Ξεκινώντας μια ιστορική αναδρομή της εξέλιξης των μεγάλων γλωσσικών μοντέλων, θα αναφερθούμε στο chatbot **«ELIZA»**, το οποίο κατασκευάστηκε το 1996 και θεωρείται το πρώτο chatbot που δημιουργήθηκε από ανθρώπους. Ο δημιουργός του, Joseph Weizenbaum, τοι ανέπτυξε στο πανεπιστήμιο του MIT. Ο τρόπος λειτουργίας του ELIZA, ήταν να δημιουργήσει την ψευδαίσθηση συνομιλίας με την τεχνική της αναδιατύπωσης δηλώσεων των χρηστών ως ερωτήσεις. Εκείνη την εποχή, δημιουργήθηκαν πολλές παραλλαγές του συγκεκριμένου chatbot οι οποίες λειτουργούσαν με παρόμοιο τρόπο και μια από τις πιο γνωστές ονομάζεται «DOCTOR», το οποίο ανταποκρινόταν σαν ψυχοθεραπευτής. Αυτή η αρχή έβαλε τις βάσεις για περεταίρω έρευνα στον τομέα των chatbots και της επεξεργασίας φυσικής γλώσσας. (Pi, 2024)

Εικόνα 5. Ένα παράδειγμα συζήτησης με το chatbot ELIZA



Σημείωση: Ανάκτηση από «ELIZA», In Wikipedia, The Free Encyclopedia (2024, July 15), (σύνδεσμος).

2.3.2 Άνοδος των νευρωνικών δικτύων

Όπως αναφέρει ο Pi(2024) στο άρθρο του, προς τα τέλη του 20ου αιώνα εμφανίστηκαν τα νευρωνικά δίκτυα, εμπνευσμένα βαθιά από τον ανθρώπινο εγκέφαλο, όπως γίνεται φανερό από την ονομασία τους αλλά και την αρχιτεκτονική τους με διασυνδεδεμένους νευρώνες. Το 1986, αναφέρεται ως η χρονιά που έκαναν την εμφάνισή τους τα «Επαναλαμβανόμενα Νευρωνικά Δίκτυα (RNN), τα οποία σε αντίθεση με τα παραδοσιακά προωθητικά νευρωνικά δίκτυα, όπου η ροή των πληροφοριών είχε μονάχα μία κατεύθυνση, μπορούσαν να θυμούνται προηγούμενες εισόδους και να απαντούν με βάση το ευρύτερο πλαίσιο. Έτσι, εκπαιδεύονταν να επεξεργάζονται και να μετατρέπουν μια ακολουθία δεδομένων εισόδου σε συγκεκριμένη ακολουθία δεδομένων εξόδου. Ωστόσο, τα RNN είχαν τον πολύ μεγάλο περιορισμό στην «μνήμη», κάτι αντίστοιχο με το σημερινό context size των μεγάλων γλωσσικών μοντέλων όπως του ChatGPT, το οποίο τα κάνει να φαίνονται σαν να «ξεχνούν» πληροφορίες από προηγούμενα μηνύματα.

2.3.3 Δημιουργία των LSTM

Το 1997 εμφανίστηκε η Μνήμη Μακράς Βραχείας Διάρκειας (LSTM), μια εξειδικευμένη μορφή RNN που βελτίωνε το πρόβλημα της διατήρησης πληροφορίας για μεγάλες ακολουθίες. Συγκεκριμένα, το εργαλείο αυτό είχε μια μοναδική αρχιτεκτονική που αποτελούνταν από πύλες εισόδου, λήθης και πύλες εξόδου, επιτρέποντας έτσι τη διατήρηση και τη διαχείριση πληροφοριών για μεγαλύτερα χρονικά διαστήματα (Pi, 2024).

2.3.4 Δημιουργία Gated Recurrent Network

Το 2014, εμφανίστηκαν οι Μονάδες Επαναλαμβανόμενων Δικτύων με Πύλες (GRU), σχεδιασμένες για να επιλύουν τα ίδια προβλήματα με τα LSTM, αλλά με απλούστερη δομή. Οι GRU χρησιμοποιούσαν δύο πύλες: μια πύλη ενημέρωσης και μια πύλη επαναφοράς. (Pi, 2024)

2.3.5 Άνοδος του συστατικού Attention

Τελικά, οι τεχνολογίες RNN, LSTM και GRU αποδείχτηκε ότι δεν είναι τόσο αποτελεσματικές στην διατήρηση των συμφραζόμενων όταν αυτά επεκτείνονται σε μεγάλο βαθμό. Έτσι, δημιουργήθηκε ο μηχανισμός που ονομάστηκε **«Attention»**, ο οποίος προσέφερε μία νέα οπτική στα μεγάλα γλωσσικά μοντέλα. Συγκεκριμένα, το attention επέτρεψε στο μοντέλο να «κοιτάει»

πίσω σε ολόκληρο το διαθέσιμο υλικό δυναμικά, επιλέγοντας τα πιο σημαντικά κομμάτια που προσθέτουν σημασία στις υπόλοιπες προτάσεις. (Pi, 2024)

Εικόνα 6. Σύγκριση της απόδοσης Attention και RNN

Σημείωση: Ανάκτηση από «Brief Introduction to the History of Large Language Models (LLMs)», W. P,2024 (σύνδεσμος)

2.3.6 Η εφεύρεση των Transformers

Το 2017 ήταν το έτος που πρωτοεμφανίστηκε η έννοια των «**Transformers»,** στο paper με τίτλο «Attention is all you need» από τον Vaswani και τους συνεργάτες του στην Google. Η νέα αυτή αρχιτεκτονική, όπως εξηγήθηκε και στην προηγούμενη υποενότητα, χρησιμοποιούσε ως βασικό της εργαλείο τον μηχανισμό attention για να επεξεργαστεί τα δεδομένα εισόδου και ήταν ικανή να επεξεργάζεται ακολουθίες παράλληλα, χωρισμένη σε πολλά στρώματα. Αυτή η προσέγγιση έθεσε τα θεμέλια για μεταγενέστερα μοντέλα όπως το ChatGPT (Pi, 2024).

2.3.7 Εμφάνιση μεγάλων γλωσσικών μοντέλων

Με την μεγάλη επιτυχία των transformers, το επόμενο λογικό βήμα ήταν η αύξηση της κλίμακας. Αυτό ξεκίνησε με το μοντέλο BERT της Google το 2018 και συνεχίστηκε με την κυκλοφορία των

GPT-2 το 2019,του GPT-3 το 2020, καθώς και των νέων εκδόσεων αυτού όπως το GPT-3.5, GPT-4 και GPT-4o (Pi, 2024).

2.4 Κύριες εφαρμογές και χρήσεις

Με την τεράστια εξέλιξη των μεγάλων γλωσσικών μοντέλων, αυτά έχουν γίνει καθημερινό εργαλείο για πολλούς ανθρώπους, εξοικονομώντας χρόνο, καθώς το μέγεθός τους από δεδομένα και η πολυπλοκότητά τους επιτρέπει σε στιγμιαίες απαντήσεις χωρίς καθυστέρηση. Το κόστος τους είναι ελάχιστο, αφού τα περισσότερα είναι διαθέσιμα δωρεάν σε απλούστερες αλλά πολύ αποτελεσματικές μορφές, ενώ η χρήση τους απαιτεί ελάχιστες γνώσεις. Όλα τα παραπάνω έχουν ωθήσει τα μεγάλα γλωσσικά μοντέλα να βρουν πολυάριθμες χρήσεις σε διάφορους τομείς. Όπως αναφέρει και ο Sumrak(2024), κάποια από τα δημοφιλέστερα παραδείγματα χρήσης των μεγάλων γλωσσικών μοντέλων περιλαμβάνουν την ανάλυση ήχου, τη δημιουργία περιεχομένου, την υποστήριξη πελατών, τη μετάφραση γλωσσών, την εκπαίδευση και την κυβερνοασφάλεια.

2.4.1 Ανάλυση ήχου

Τα μεγάλα γλωσσικά μοντέλα έχουν αναθεωρήσει τον τρόπο με τον οποίο οι άνθρωποι χειρίζονται τα δεδομένα ήχου. Συγκεκριμένα, έχουν την ικανότητα να ακούν πολύωρες συζητήσεις και να παράγουν αποτελεσματικές περιλήψεις καθώς και να απαντούν ερωτήσεις σχετικά με μακροσκελείς συναντήσεις. Ακόμη, μπορούν, με βάση ένα μεγάλο ποσοστό κλήσεων ως δεδομένα, να εξάγουν πολύπλοκα αποτελέσματα και συμβουλές βελτίωσης (Sumrak, 2024).

2.4.2 Δημιουργία περιεχομένου

Τα μεγάλα γλωσσικά μοντέλα χρησιμοποιούνται αποτελεσματικά από συγγραφείς και εμπόρους για την δημιουργία αρχικών σχεδίων (drafts), την πρόταση διάφορων αλλαγών και για την ταχεία εύρεση άρθρων και αναφορών στο διαδίκτυο. Αυτά τα εργαλεία επιταχύνουν σημαντικά την παραγωγικότητα, επιτρέποντάς στους χρήστες να επικεντρωθούν παραπάνω στο πιο απαιτητικό και δημιουργικό κομμάτι της δουλείας τους, αφήνοντας τα μοντέλα να ασχοληθούν με τα μηχανικά στοιχεία της εργασίας τους (Sumrak, 2024).

2.4.3 Υποστήριξη πελατών

Ένας τομέας στον οποίο τα μεγάλα γλωσσικά μοντέλα έχουν κάνει ραγδαία άνοδο και χρησιμοποιούνται κατά κόρον, είναι η εξυπηρέτηση πελατών. Πολλές εταιρείες τηλεφωνίας, σελίδες του δημοσίου τομέα και μεγάλες επιχειρήσεις έχουν υιοθετήσει μοντέλα εξυπηρέτησης

πελατών μέσω μεγάλων γλωσσικών μοντέλων. Αυτά είναι διαθέσιμα 24/7 χωρίς την ανάγκη ανθρώπινης παρέμβασης, προσφέροντας συνεχή βοήθεια σε χρήστες από όλο τον κόσμο με πολύ χαμηλό κόστος (Sumrak, 2024).

2.4.4 Μετάφραση γλωσσών

Τα μεγάλα γλωσσικά μοντέλα βοηθούν στην άρση των γλωσσικών φραγμών, δίνοντας την δυνατότητα στις επιχειρήσεις να προσεγγίζουν πελάτες αλλά και να προσλαμβάνουν άτομα από όλες τις χώρες. Αυτά τα μοντέλα προσφέρουν ακριβείς υπηρεσίες μετάφρασης σε πραγματικό χρόνο, κάνοντας ιστότοπους, εφαρμογές και ψηφιακό περιεχόμενο παγκοσμίως προσβάσιμα (Sumrak, 2024).

2.4.5 Εκπαίδευση

Στον τομέα της εκπαίδευσης, τα μεγάλα γλωσσικά μοντέλα χρησιμοποιούνται για την παροχή εξατομικευμένης μάθησης προσαρμόζοντας το περιεχόμενο στις ατομικές ανάγκες των μαθητών, δημιουργώντας ερωτήσεις κατανόησης και παρέχοντας λεπτομερείς εξηγήσεις προσαρμοσμένες στις ανάγκες των μαθητών (Sumrak, 2024).

2.4.6 Κυβερνοασφάλεια

Τα μεγάλα γλωσσικά μοντέλα μπορούν να χρησιμοποιηθούν στην ανάλυση και την ερμηνεία μεγάλων ποσών δεδομένων κυβερνοασφάλειας, προβλέποντας, αναγνωρίζοντας και ανταποκρινόμενα σε πιθανές απειλές ασφαλείας. Η στοχευμένης εκπαίδευσή τους επιτρέπει ταχύτερη και πιο ακριβή ανίχνευση και ανταπόκριση στις απειλές, ενισχύοντας την ασφάλεια των επιχειρήσεων (Sumrak, 2024).

2.5 Παραδείγματα Μεγάλων Γλωσσικών Μοντέλων

Ολοκληρώνοντας την ανάλυση των μεγάλων γλωσσικών μοντέλων, είναι σημαντικό να αναφερθούν κάποια από τα πιο πολυχρησιμοποιούμενα μοντέλα και οι κυριότερες λειτουργίες και διαφοροποιήσεις τους.

2.5.1 BERT

Το πρώτο μεγάλο γλωσσικό μοντέλο που αναπτύχθηκε ήταν το **BERT** (Bidirectional Encoder Representations from Transformers), δημιουργημένο από την Google το έτος 2018. Το BERT είναι ένα μοντέλο που χρησιμοποιεί την αρχιτεκτονική των transformers με πολυάριθμα στρώματα

συνδεδεμένα μεταξύ τους και διαθέτει 342 εκατομμύρια παραμέτρους για την επεξεργασία εισόδων. Εκπαιδεύτηκε σε τεράστιο όγκο δεδομένων για να παράγει απαντήσεις σε φυσική γλώσσα που να είναι κατανοητές από τους ανθρώπους (Lutkevich, 2024).

2.5.2 GEMINI

Ακολουθώντας την Google, το ανανεωμένο μοντέλο της ονομάστηκε **GEMINI**, το οποίο έχει την ονομασία του από το chatbot της εταιρείας. Το GEMINI είναι ένα πολυτροπικό μοντέλο, ικανό να επεξεργάζεται ήχο, εικόνα, βίντεο και κείμενο, σε αντίθεση με άλλα γλωσσικά μοντέλα που περιορίζονται μόνο σε κείμενο. Το GEMINI διατίθεται σε τρία «μεγέθη», το Ultra, το Pro και το Nano, από το μεγαλύτερο και πιο ικανό έως το μικρότερο και λιγότερο ικανό. Υπάρχουν αναφορές ότι το GEMINI υπερτερεί σε δύναμη από το μοντέλο GPT -4 της OpenAI, στο οποίο θα γίνει αναφορά στην συνέχεια. (Lutkevich, 2024).

2.5.3 GPT –3

Το **GPT -3**(Generative Pre-trained Transformer 3) αποτελεί το πρώτο ισχυρό μοντέλο της OpenAI το οποίο παρουσιάστηκε το 2020 με περισσότερες από 175 δισεκατομμύρια παραμέτρους (το BERT έχει 342 εκατομμύρια). Ενσωματώνει την αρχιτεκτονική των συνδεδεμένων στρωμάτων transformers και είναι δέκα φορές μεγαλύτερο από τον προκάτοχο του, το GPT -2. Είναι εκπαιδευμένο με εκατομμύρια δεδομένα και αποτελεί το τελευταίο μοντέλο από την σειρά μοντέλων που παρήγαγε η OpenAI, για το οποίο ήταν δημόσια γνωστός ο αριθμός των παραμέτρων που χρησιμοποιούσε (Lutkevich, 2024).

2.5.4 GPT -3.5 και GPT –3.5 Turbo

Το **GPT –3.5** αποτελεί την ενημερωμένη έκδοση του GPT –3 με λιγότερες παραμέτρους αλλά ποιοτικότερη εκπαίδευση, και διαδραμάτισε κρίσιμο ρόλο στην πλατφόρμα ChatGPT, η οποία το 2023 άλλαξε ριζικά τη δημοτικότητα και χρήση των μεγάλων γλωσσικών μοντέλων. Το GPT –3.5 εκπαιδεύτηκε με γνώσεις μέχρι και τον Σεπτέμβρη του 2021 και δεν έχει την δυνατότητα πρόσβασης στο διαδίκτυο συγκριτικά με άλλα μεγάλα γλωσσικά μοντέλα Παράλληλα, η πιο ισχυρή έκδοση του GPT-3.5 είναι το **GPT-3.5 Turbo**, που χρησιμοποιείται από το GitHub Copilot της Microsoft για την υποστήριξη των προγραμματιστών με παραγωγή κώδικα και ανίχνευση σφαλμάτων (Lutkevich, 2024).

2.5.5 GPT –4

Το **GPT-4** αποτελεί το μεγαλύτερο μοντέλο της σειράς GPT της OpenAI, κυκλοφορώντας το 2023. Όπως και τα προηγούμενα μοντέλα, βασίζεται στην αρχιτεκτονική των transformers. Η OpenAI περιγράφει το GPT-4 ως ένα πολυτροπικό μοντέλο(όπως και το GEMINI), πράγμα που σημαίνει ότι είναι ικανό να επεξεργάζεται κείμενο, ήχο και εικόνα, σε αντίθεση με τα προηγούμενα μοντέλα της OpenAI που περιορίζονταν μόνο στο κείμενο. Το GPT-4, υποστηρίζεται ότι πλησίασε την τεχνητή γενική νοημοσύνη (AGI), που σημαίνει ότι είναι εξίσου έξυπνο ή εξυπνότερο από έναν άνθρωπο και σημαντικότερα έχει την δυνατότητα της πρόσβασης στο διαδίκτυο, σε αντίθεση με το GPT –3.5 (Lutkevich, 2024).

2.5.6 GPT –4o

Το **GPT -4o (**GPT -4 Omni)είναι ο διάδοχος του GPT-4 και το πιο πρόσφατο μεγάλο γλωσσικό μοντέλο της OpenAI . Προσφέρει αρκετές βελτιώσεις σε σχέση με το GPT -4 με μια πιο φυσική ανθρώπινη αλληλεπίδραση για το ChatGPT. Το GPT-4o είναι επίσης πολυτροπικό, αλλά με τη δυνατότητα να βλέπει φωτογραφίες ή οθόνες και να κάνει σχετικές ερωτήσεις κατά τη διάρκεια της αλληλεπίδρασης.

3. Μεγάλα Γλωσσικά Μοντέλα στην Ανάπτυξη Λογισμικού

3.1 Ρόλος των μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού

3.1.1 Εισαγωγή στην έννοια των γλωσσικών μοντέλων στην ανάπτυξη λογισμικού

Τα μεγάλα γλωσσικά μοντέλα είναι προηγμένα συστήματα τεχνητής νοημοσύνης που ανταποκρίνονται σε ερωτήσεις και συζητήσεις χρηστών με απαντήσεις σε ανθρώπινη γλώσσα. Έχουν την ικανότητα να κατανοούν βαθιά τα συμφραζόμενα και τις ανάγκες του χρήστη, κάνοντάς τα χρήσιμα σε πολλούς τομείς της καθημερινότητας και ιδιαίτερα στην ανάπτυξη λογισμικού. Τα γλωσσικά μοντέλα μπορούν να βοηθήσουν προγραμματιστές, αυξάνοντας την παραγωγικότητά τους και μειώνοντας τον χρόνο εκσφαλμάτωσης, καθώς έχουν εκπαιδευτεί σε δισεκατομμύρια έργα ανοιχτού κώδικα και ενημερώνονται συνεχώς με νέα δεδομένα.

Ο εσωτερικός τρόπος λειτουργίας των μεγάλων γλωσσικών μοντέλων τα καθιστά εξαιρετικά στην υποβοήθηση πάρα πολλών έργων και στην λύση των δυσκολιών που μπορεί να προκύψουν για έναν προγραμματιστή κατά την διάρκεια της διαδικασίας της ανάπτυξης λογισμικού, αυξάνοντας έτσι σημαντικά την παραγωγικότητα, μειώνοντας τον χρόνο εκσφαλμάτωσης και συνεπώς και τον χρόνο που απαιτείται για την ολοκλήρωση ενός έργου. Όλα αυτά οφείλονται στο γεγονός ότι τα μεγάλα γλωσσικά μοντέλα έχουν την ικανότητα να γράφουν, να διορθώνουν και να βελτιστοποιούν τον κώδικα πολύ γρήγορα και με μεγάλη ακρίβεια, καθώς έχουν εκπαιδευτεί σε δισεκατομμύρια έργα ανοιχτού κώδικα, όπως αυτά στο GitHub, ενώ παράλληλα ενημερώνονται συνεχώς με νέα δεδομένα.

3.1.2 Παραδείγματα χρήσης μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού

Ορισμένα από τα σημαντικότερα παραδείγματα χρήσης των μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού περιλαμβάνουν την ανάλυση κώδικα, την αυτόματη παραγωγή ελέγχων του κώδικα, την ανατροφοδότηση προγραμματιστή κατά τη συγγραφή κώδικα καθώς και τεκμηρίωση του κώδικα.

Η ανάλυση κώδικα μπορεί να εξοικονομήσει πολύ χρόνο και να μειώσει τα λάθη, καθώς τα γλωσσικά μοντέλα μπορούν να δημιουργούν κώδικα βάσει των περιγραφών των προγραμματιστών και να προτείνουν βελτιώσεις. Η αυτόματη παραγωγή ελέγχων εξοικονομεί χρόνο και ελαχιστοποιεί τα ανθρώπινα λάθη, ενώ εντοπίζει όλες τις ακραίες περιπτώσεις ελέγχου του συστήματος. Η ανατροφοδότηση κατά την συγγραφή επιτρέπει στους προγραμματιστές να

εντοπίζουν αμέσως λάθη και να διορθώνουν σφάλματα γρήγορα και αποτελεσματικά και τεκμηρίωση του κώδικα βελτιώνει την κατανοησιμότητα και την συντηρησιμότητα του κώδικα (Ozkaya και συν., 2023).

3.1.3 Ανάλυση κώδικα

Στον τομέα της ανάλυσης κώδικα, τα μεγάλα γλωσσικά μοντέλα μπορούν να αναλύουν κώδικα σε μεγάλη κλίμακα με υψηλή ταχύτητα και ακρίβεια, βοηθώντας τους προγραμματιστές να εντοπίζουν σύνθετα προβλήματα που μπορεί να είναι δύσκολο να εντοπιστούν. Η ικανότητα των μεγάλων γλωσσικών μοντέλων να επεξεργάζονται μεγάλες ποσότητες κώδικα τα καθιστά εξαιρετικά χρήσιμα εργαλεία για την αναγνώριση ασυμφωνιών και ελλείψεων. Τα μεγάλα γλωσσικά μοντέλα μπορούν να προτείνουν βελτιώσεις και διορθώσεις, επιταχύνοντας τη διαδικασία ανάπτυξης και μειώνοντας τον κίνδυνο λαθών που θα μπορούσαν να επηρεάσουν τη λειτουργικότητα και την απόδοση της εφαρμογής (Ozkaya και συν., 2023).

3.1.4 Αυτόματη παραγωγή ελέγχων κώδικα

Τα μεγάλα γλωσσικά μοντέλα μπορούν να αυτοματοποιήσουν την παραγωγή των ελέγχων του κώδικα, κάτι το οποίο μπορεί να συμβάλλει σημαντικά στη βελτίωση της ποιότητας του κώδικα και στην αποδοτικότητα της διαδικασίας ανάπτυξης. Τα μοντέλα αυτά μπορούν να δημιουργούν ελέγχους με βάση προκαθορισμένα πρότυπα και απαιτήσεις, αναγνωρίζοντας αυτόματα περιοχές του κώδικα που χρειάζονται επιπλέον δοκιμές ή επανεξέταση. Αυτό μειώνει την ανάγκη για χειροκίνητη δημιουργία ελέγχων, μειώνοντας την πιθανότητα ανθρώπινου λάθους και επιτρέποντας στους προγραμματιστές να εστιάσουν σε πιο στρατηγικές και δημιουργικές πτυχές της ανάπτυξης λογισμικού (Ozkaya και συν., 2023).

3.1.5 Ανατροφοδότηση κατά τη συγγραφή κώδικα

Τα εργαλεία που βασίζονται σε μεγάλα γλωσσικά μοντέλα, όπως το GitHub Copilot, παρέχουν ανατροφοδότηση σε πραγματικό χρόνο κατά τη συγγραφή κώδικα, συμβάλλοντας στη βελτίωση της αποτελεσματικότητας των προγραμματιστών. Αυτή η ανατροφοδότηση περιλαμβάνει προτάσεις για τη βελτίωση του συντακτικού, διόρθωση λογικών σφαλμάτων και προτάσεις για την ορθή δημιουργία κώδικα. Η ικανότητα των γλωσσικών μοντέλων να προσαρμόζονται στις ιδιαίτερες ανάγκες και στυλ κωδικοποίησης των προγραμματιστών ενισχύει τη διαδικασία ανάπτυξης, μειώνοντας την πιθανότητα λαθών και βελτιώνοντας τη συνολική ποιότητα του κώδικα. Αυτή η υποστήριξη σε πραγματικό χρόνο επιτρέπει στους προγραμματιστές να

εργάζονται πιο αποδοτικά και να διασφαλίζουν ότι ο κώδικας τους πληροί τα απαιτούμενα πρότυπα (Ozkaya και συν., 2023).

3.1.6 Τεκμηρίωση κώδικα

Ένα ακόμη παράδειγμα της χρήση των μεγάλων γλωσσικών μοντέλων στην διαδικασία της ανάπτυξης λογισμικού είναι στη διαδικασία τεκμηρίωσης κώδικα, η οποία γίνεται πιο αποτελεσματική και ακριβής με τη χρήση των μεγάλων γλωσσικών μοντέλων. Τα μοντέλα μπορούν να δημιουργούν αυτόματα σχόλια και τεκμηρίωση για τον κώδικα, βασισμένα σε αναλύσεις του ίδιου του κώδικα και των γύρω εγγράφων. Αυτό βοηθά στη βελτίωση της αναγνωσιμότητας του κώδικα και της συντηρησιμότητάς του, εξασφαλίζοντας ότι η τεκμηρίωση είναι πάντα ενημερωμένη και ακριβής. Επιπλέον, τα γλωσσικά μοντέλα μπορούν να συνοψίζουν και να παρέχουν απαντήσεις σε ερωτήσεις σχετικά με έγγραφα προδιαγραφών ή πολιτικές, διευκολύνοντας την κατανόηση και εφαρμογή των απαιτήσεων του έργου (Ozkaya και συν., 2023).

3.2 Χρήσεις σε διάφορα στάδια της ανάπτυξης λογισμικού

3.2.1 Στάδιο ανάλυσης απαιτήσεων

Η ανάλυση των απαιτήσεων είναι μία κρίσιμη διαδικασία για την επιτυχή ανάπτυξη ενός συστήματος λογισμικού. Ένα σύνηθες πρόβλημα κατά το στάδιο της ανάλυσης των απαιτήσεων, είναι η ύπαρξη ασαφειών, δηλαδή η διαφορετική ερμηνεία της ίδιας απαίτησης από πολλά άτομα, το οποίο μπορεί να οδηγήσει σε σοβαρά προβλήματα στα υπόλοιπα στάδια της ανάπτυξης λογισμικού. Σύμφωνα με τον Hou και συν. (2023), το ChatGPT, υπερτερεί σε μεγάλο βαθμό σε σχέση με τα άλλα γλωσσικά μοντέλα. Παράλληλα, όπως ανέδειξε και μια έρευνα που αναφέρει ο Hou και οι συν. στην οποία δόθηκαν κάποιες απαιτήσεις με μεγάλες προκλήσεις στην ασάφεια, το γλωσσικό μοντέλο κατάφερε να εντοπίσει με ακρίβεια τα λάθη και να τα διορθώσει σε κάθε περίπτωση. Επομένως, τα μεγάλα γλωσσικά αποτελούν ένα πολύ σημαντικό εργαλείο στην εξάλειψη των ασαφειών στο στάδιο της ανάλυσης απαιτήσεων, συμβάλλοντας σε ένα ποιοτικότερο αποτέλεσμα.

3.2.2 Στάδιο σχεδιασμού

Στο στάδιο του σχεδιασμού είναι επίσης κρίσιμο σημείο για την επιτυχία ενός έργου ανάπτυξης λογισμικού. Όπως αναφέρει ο Hou και συν.(2023), ενώ η έρευνα γύρω από τα μεγάλα γλωσσικά

μοντέλα στον τομέα του σχεδιασμού δεν έχει προχωρήσει σε βάθος σε σύγκριση με άλλα στάδια της ανάπτυξης, όπως η κωδικοποίηση και η βελτιστοποίηση, τα μεγάλα γλωσσικά μοντέλα προσφέρουν σημαντική υποστήριξη με ποικίλους άλλους τρόπους. Ειδικότερα, τα γλωσσικά μοντέλα μπορούν να βελτιώσουν την υπάρχουσα αρχιτεκτονική και να προτείνουν διάφορες εναλλακτικές λύσεις, εάν αυτό κριθεί απαραίτητο. Για παράδειγμα, μπορούν να προτείνουν συγκεκριμένα πρότυπα σχεδίασης που να ταιριάζουν περισσότερο με τις ανάγκες του έργου.

Ακόμη, τα μεγάλα γλωσσικά μοντέλα μπορούν να συνεισφέρουν στην δημιουργία διαγραμμάτων, όχι μέσω της παραγωγής τους απευθείας αλλά μέσω της περιγραφής μέσω μηνυμάτων, στις περιγραφές συστημάτων και στις τεχνικές προδιαγραφές.

Επιπρόσθετα, τα μεγάλα γλωσσικά μοντέλα μπορούν να εντοπίζουν σημεία όπου μπορεί να υπάρξουν πιθανά σφάλματα απόδοσης ή ασφάλειας και να προτείνουν βελτιώσεις, συμβάλλοντας σε μία αποδοτικότερη και ασφαλέστερη αρχιτεκτονική.

3.2.3 Στάδιο κωδικοποίησης

Το στάδιο της κωδικοποίησης είναι το πιο γνωστό και συχνά το πιο χρονοβόρο στάδιο στην ανάπτυξη λογισμικού. Σε αυτό το στάδιο, οι προγραμματιστές γράφουν τον κώδικα που θα υλοποιήσει τις απαιτήσεις και τις προδιαγραφές του έργου τους. Τα μεγάλα γλωσσικά μοντέλα παρουσιάζουν συνεχώς βελτιωμένες επιδόσεις με κάθε νέο μοντέλο που κυκλοφορεί. Όπως αναφέρει ο Hou και συν. (2023), μοντέλα όπως τα GPT-4, GPT-3, GPT-3.5, BERT series, Codex, CodeGen, InCoder, Copilot και CodeGeeX έχουν πολύ σημαντικό ρόλο στην κωδικοποίηση. Η εκπαίδευσή τους σε μεγάλους όγκους δεδομένων και συγκεκριμένα κειμένων, τους επιτρέπει να κατανοούν βαθιά τη φυσική γλώσσα και να τη μετατρέπουν σε κώδικα, ενώ παράλληλα τα μοντέλα αποδίδουν εξαιρετικά στην παραγωγή κώδικα με περιγραφές σε φυσική γλώσσα, προτείνοντας διορθώσεις και συμπληρώσεις σε πραγματικό χρόνο.

Ακόμη, τα μοντέλα αυτά συνεισφέρουν και στην δημιουργία τεκμηρίωσης, ένα σημαντικό αλλά συχνά παραμελημένο κομμάτι της ανάπτυξης λογισμικού. Τα γλωσσικά μοντέλα μπορούν να παράγουν αυτόματα σχόλια και τεκμηρίωση για τον κώδικα, διευκολύνοντας την κατανόηση και τη συντήρηση του κώδικα από άλλους προγραμματιστές.

Επιπλέον, τα μεγάλα γλωσσικά μοντέλα υποστηρίζουν πολλές γλώσσες προγραμματισμού, καθιστώντας τα χρήσιμα για πολυγλωσσικά έργα που απαιτούν χρήση πολλών γλωσσών, εφόσον

μπορούν να αναγνωρίζουν τα συμφραζόμενα και να προσαρμόζουν τις προτάσεις τους ανάλογα με τη γλώσσα προγραμματισμού που χρησιμοποιείται κάθε φορά (Zharovskikh, 2023)

3.2.4 Στάδιο ελέγχου και βελτιστοποίησης

Το στάδιο του ελέγχου και της βελτιστοποίησης είναι ζωτικής σημασίας για τη διασφάλιση της ποιότητας και της απόδοσης του λογισμικού. Κατά τη διάρκεια αυτού του σταδίου, οι προγραμματιστές και οι δοκιμαστές εντοπίζουν και διορθώνουν σφάλματα, βελτιστοποιούν την απόδοση και διασφαλίζουν ότι το λογισμικό πληροί τις απαιτήσεις των χρηστών. Ο έλεγχος και η βελτιστοποίηση διαρκούν συνήθως αρκετό καιρό, καθώς απαιτείται λεπτομερής έλεγχος όλων των κομματιών κώδικα που έχουν δημιουργηθεί. Επιπλέον, η διαδικασία αυτή συνεχίζεται και μετά την παράδοση του τελικού προϊόντος, εφόσον προκύψουν νέες απαιτήσεις από τους χρήστες και τον εμπλεκόμενους του έργου.

Τα μεγάλα γλωσσικά μοντέλα προσφέρουν πολύτιμη υποστήριξη σε αυτό το στάδιο, μέσω της δημιουργίας αυτοματοποιημένων τεστ, τον εντοπισμό σφαλμάτων και την βελτιστοποίηση του κώδικα.

Συγκεκριμένα, τα μεγάλα γλωσσικά μοντέλα μπορούν να δημιουργούν αυτόματα μονάδες ελέγχου (unit tests) και ενσωματωμένα τεστ (integration tests) βάση του κώδικα που έχει γραφτεί. Με αυτόν τον τρόπο εξασφαλίζεται ότι κάθε κομμάτι κώδικα λειτουργεί σωστά και αποδοτικά.

Ακόμη, τα μεγάλα γλωσσικά μοντέλα μπορούν να αναλύουν μεγάλα κομμάτια κώδικα και να εντοπίζουν πιθανά συντακτικά και λογικά σφάλματα που έχουν συμβεί. Παρέχουν επίσης προτάσεις για τη διόρθωση αυτών των σφαλμάτων, διευκολύνοντας τους προγραμματιστές στη διαδικασία της εκσφαλμάτωσης.

Ακόμη, τα μοντέλα αυτά έχουν την δυνατότητα να εντοπίζουν αναποτελεσματικά κομμάτια κώδικα που μπορεί να δίνουν το επιθυμητό αποτέλεσμα, αλλά όχι με τον βέλτιστο τρόπο, και να προτείνουν βέλτιστες πρακτικές και βελτιώσεις, οδηγώντας σε πιο ποιοτικό κώδικα.

3.3 Πλεονεκτήματα και περιορισμοί

Η χρήση των μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού προσφέρει πληθώρα πλεονεκτημάτων και νέες δυνατότητες στους προγραμματιστές, με ορισμένους όμως περιορισμούς στην ορθή χρήση και τον έλεγχο των αποτελεσμάτων.

3.3.1 Πλεονεκτήματα

Καταρχάς, η ικανότητα των μεγάλων γλωσσικών μοντέλων να δημιουργούν με τόσο μεγάλη ταχύτητα κώδικα βελτιώνει την αποδοτικότητα και μειώνει τα σφάλματα κατά τη διαδικασία της κωδικοποίησης, ενώ παράλληλα λειτουργεί και ως ένα εκπαιδευτικό εργαλείο για τον προγραμματιστή, αφού προσφέρει άμεσα εξηγήσεις και απαντήσεις σχετικά με τις διορθώσεις του κώδικα (Hurani & Idris, 2024).

Ακόμη, μέσω των διορθώσεων και των συμπληρώσεων σε πραγματικό χρόνο που προσφέρουν αυτά τα μοντέλα, επιταχύνουν την ανάπτυξη λογισμικού και διασφαλίζουν υψηλή ποιότητα στον παραγόμενο κώδικα.

Επιπλέον, η δυνατότητά τους να υποστηρίζουν πολλές γλώσσες προγραμματισμού τα καθιστά ιδιαίτερα χρήσιμα για πολυγλωσσικά έργα, όπου η κατανόηση μεγάλου όγκου κώδικα σε διαφορετικές γλώσσες προγραμματισμού μπορεί να είναι μία πρόκληση για τους εμπλεκόμενους προγραμματιστές. (Zharovskikh, 2023)

Τέλος, η εκπαίδευσή των μεγάλων γλωσσικών μοντέλων σε μεγάλα σύνολα δεδομένων και projects, επιτρέπει βαθιά κατανόηση και εφαρμογή των βέλτιστων τεχνικών της ανάπτυξης λογισμικού, κάνοντάς τα πολύτιμα εργαλεία για προγραμματιστές και μεγάλες ομάδες ανάπτυξης.

3.3.2 Περιορισμοί

Σε συνδυασμό με τα πλεονεκτήματα, η εκτεταμένη χρήση των μεγάλων γλωσσικών μοντέλων κατά την διαδικασία της ανάπτυξης λογισμικού συνεπάγεται και ορισμένους περιορισμούς.

Συγκεκριμένα, ενώ τα μεγάλα γλωσσικά μοντέλα διαπρέπουν σε απλά σενάρια και έργα, σε πολλές περιπτώσεις παρουσιάζουν περιορισμένη ακρίβεια σε πιο εξειδικευμένα ή σύνθετα σενάρια, λόγω της πιθανής ελλιπής εκπαίδευσής τους σε παρόμοια δεδομένα, με αποτέλεσμα την παραγωγή ανεπαρκών αποτελεσμάτων. (Hurani & Idris, 2024)

Ακόμη, όπως αναφέρουν οι Hurani και Idris (2024), η χρήση κώδικα η οποία έχει παραχθεί από μεγάλα γλωσσικά μοντέλα, αυξάνει την ανάγκη για επαλήθευση και διορθώσεις από τους προγραμματιστές, καθώς το μοντέλο ενδέχεται να μην κατανοεί πλήρως τις απαιτήσεις ή να παράγει κάτι παραπλήσιο από αυτό που ζητήθηκε. Αυτό μπορεί να αυξήσει σημαντικά τον χρόνο διόρθωσης σφαλμάτων, σε σύγκριση με τη συγγραφή κώδικα από τους ίδιους τους προγραμματιστές.

Τέλος, από τον τρόπο κατασκευής τους, τα μεγάλα γλωσσικά μοντέλα εξαρτώνται άμεσα από το σύνολο των δεδομένων όπου έχουν εκπαιδευτεί, με αποτέλεσμα η ποιότητα και το περιεχόμενο των δεδομένων να επηρεάζουν σε μεγάλο βαθμό τα παραγόμενα αποτελέσματα.

3.4 Μελλοντικές προοπτικές και ευκαιρίες ανάπτυξης

Τα μεγάλα γλωσσικά μοντέλα, ήδη ισχυρά εργαλεία στην ανάπτυξη λογισμικού, συνεχώς εξελίσσονται, με νέες και υποσχόμενες εφαρμογές και βελτιώσεις να μπορούν να επεκτείνουν περαιτέρω τις δυνατότητές τους. Με τις βελτιώσεις αυτές, στις οποίες θα γίνει αναφορά στην συνέχεια, αναμένεται ότι τα μοντέλα αυτά θα προσφέρουν ακόμη πιο προηγμένα εργαλεία και λειτουργικότητες για να κάνουν χρήση οι προγραμματιστές.

3.4.1 Σύμπραξη μεγάλων γλωσσικών μοντέλων

Μία υποσχόμενη προοπτική η οποία θα άξιζε να ερευνηθεί σε μεγαλύτερο βαθμό, όπως αναφέρει και ο Hou και συν.(2023), είναι η ενσωμάτωση πολλαπλών διαφορετικών μεγάλων γλωσσικών μοντέλων σε ένα ενιαίο σύστημα, όπου τα μοντέλα θα συνεργάζονται για την επίλυση σύνθετων εργασιών στην ανάπτυξη λογισμικού. Αυτή η σύμπραξη θα αξιοποιούσε τα διαφορετικά πλεονεκτήματα κάθε μοντέλου, δημιουργώντας ένα εξαιρετικά αποτελεσματικό σύστημα.

3.4.2 Κατανόηση διαφορετικού τύπου εισόδων

Επιπρόσθετα, μια σημαντική ευκαιρία για την ανάπτυξη των μεγάλων γλωσσικών μοντέλων στον τομέα της ανάπτυξης λογισμικού, είναι η βελτίωση της κατανόησης διαφορετικών τύπων εισόδου, όπως εικόνες, σχήματα και φωνή, καθώς και η αυτόματη παραγωγή πολύπλοκων διαγραμμάτων όπως UML. Παρόλο που πολυτροπικά μοντέλα, όπως το GPT-4, έχουν σημειώσει πρόοδο σε αυτούς τους τομείς, δεν έχουν ακόμα φτάσει σε ένα πλήρως υποσχόμενο επίπεδο ώστε να αποτελούν ένα έμπιστο εργαλείο για την υλοποίηση των παραπάνω(Hou και συν., 2023).

3.4.3 Προσαρμογή στις ανάγκες των χρηστών

Επιπλέον, όπως αναφέρει και η OpenAI, μία πολύ σημαντική εξέλιξη αναμένεται να ενσωματωθεί στα ήδη μεγάλα γλωσσικά μοντέλα της, τα οποία θα διαθέτουν πιο προηγμένες ικανότητες μάθησης και προσαρμογής, επιτρέποντάς τους να μαθαίνουν από τις αλληλεπιδράσεις με τους χρήστες και να προσαρμόζονται στις συγκεκριμένες ανάγκες και προτιμήσεις τους. Αυτές οι δυνατότητες θα μπορούσαν να βελτιώσουν την ακρίβεια των προτάσεων κώδικα και να

προσφέρουν εξατομικευμένες λύσεις που ανταποκρίνονται καλύτερα στις απαιτήσεις των έργων, αφού το μεγάλο γλωσσικό μοντέλο θα επικεντρώνεται σε συγκεκριμένους τομείς.

3.4.4 Βελτίωση στις ήδη υπάρχουσες εργασίες

Ωστόσο, είναι εξίσου σημαντικό τα μεγάλα γλωσσικά μοντέλα να συνεχίσουν να βελτιώνονται και στις ήδη υπάρχουσες εργασίες που εκτελούν καλά, μέσω της εκπαίδευσής τους σε περισσότερα και ποιοτικότερα δεδομένα. Αυτό περιλαμβάνει την περαιτέρω βελτίωση της ικανότητάς τους να παράγουν κώδικα με βάση τις απαιτήσεις σε φυσική γλώσσα, την ακριβή πρόταση διορθώσεων και συμπληρώσεων και τη μείωση των σφαλμάτων κατά τη διάρκεια της ανάπτυξης. Η ενίσχυση σε αυτούς τους τομείς θα βελτιώσει τη συνολική αποδοτικότητα των εργαλείων αυτών και θα συμβάλει στην ακόμα καλύτερη ενσωμάτωσή τους και αποδοχή στη διαδικασία ανάπτυξης λογισμικού.

4. Behavior-Driven Development (BDD)

Το κεφάλαιο αυτό εστιάζει στη μεθοδολογία του **Behavior Driven Development** (BDD), μια εξέλιξη του Test Driven Development (TDD) που στοχεύει στην προώθηση της συνεργασίας μεταξύ προγραμματιστών, ειδικών ελέγχου ποιότητας και πελατών. Η μεθοδολογία αυτή ενισχύει τη συνεργασία και την επικοινωνία μεταξύ των μελών μιας ομάδας ανάπτυξης λογισμικού, βελτιώνοντας έτσι την ποιότητα και τη σαφήνεια των απαιτήσεων και της λειτουργικότητας του λογισμικού.

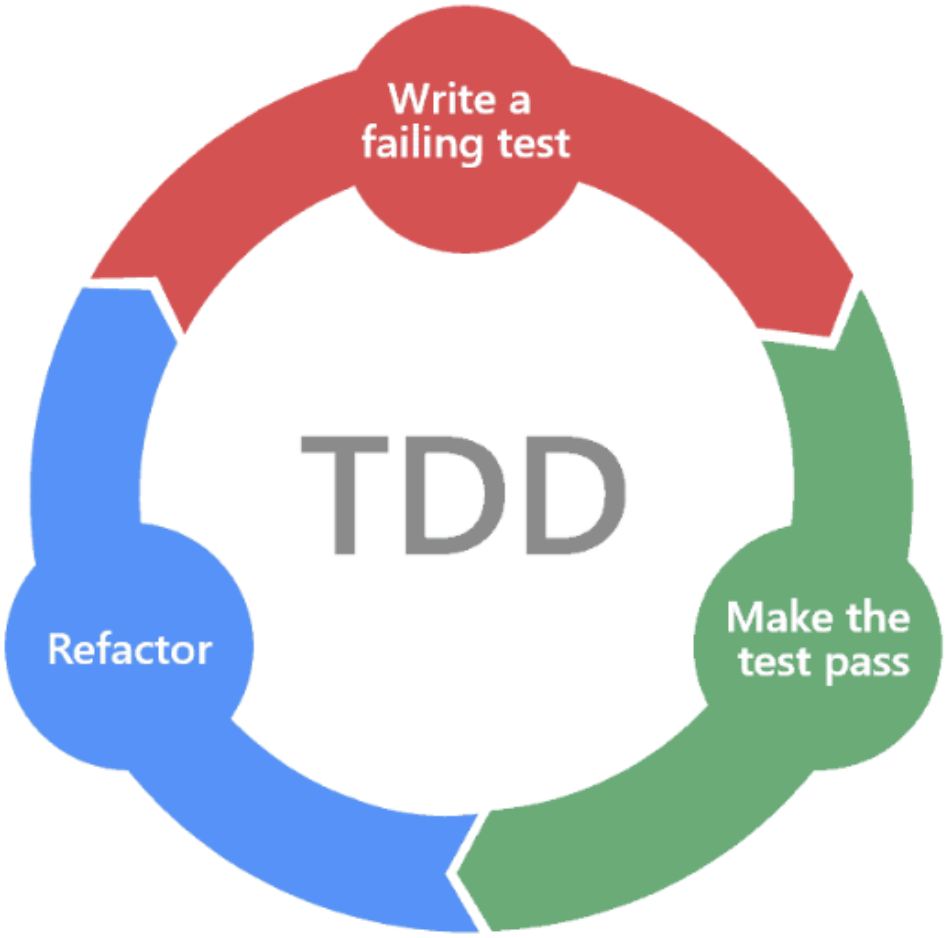
4.1 Test Driven Development και Behavior Driven Development

Ξεκινώντας το κεφάλαιο, θα γίνει μία συνοπτική αναφορά στην μεθοδολογία TDD και τις αρχές τις, έτσι ώστε στην συνέχεια να γίνει κατανοητή η έννοια του BDD που αποτελεί μία εξέλιξη αυτής.

Το **Test Driven Development (TDD)** είναι μία μεθοδολογία ανάπτυξης λογισμικού που δίνει έμφαση στην δημιουργία δοκιμών (tests) πριν αρχίσει η διαδικασία της κωδικοποίησης, κάτι το οποίο φαίνεται αντίθετο με τις παραδοσιακές πρακτικές, που τοποθετούν τις δοκιμές σαν το τελευταίο στάδιο στην διαδικασία ανάπτυξης λογισμικού.

Το TDD υποστηρίζει ότι οι δοκιμές αρχικά σχεδιάζονται με την δεδομένη συνθήκη ότι θα αποτύχουν, αφού δεν υπάρχει ακόμη ο απαραίτητος κώδικας για να περάσουν οι έλεγχοι. Στη συνέχεια, αναπτύσσεται ο κώδικας της νέας μονάδας ή λειτουργικότητας με την πιο απλή λύση ώστε απλά να περάσει το τεστ. Μετά την υλοποίηση, οι δοκιμές εκτελούνται ξανά για να επαληθευτεί ότι η υλοποίηση καλύπτει τις απαιτήσεις των δοκιμών και τέλος θα πρέπει να γίνουν οι απαραίτητες τροποποιήσεις για να βελτιωθεί η ποιότητα του υλοποιημένου ελέγχου της μονάδας (Test-driven development, 2024).

Εικόνα 7. Ο κύκλος ανάπτυξης του TDD



Σημείωση: Ανάκτηση από «Why Test-Driven Development (TDD)», MARNSER,(σύνδεσμος)

Το κλειδί στο TDD είναι ότι η κάλυψη του κώδικα δεν είναι ο τελικός στόχος, αλλά το αποτέλεσμα της σωστής εφαρμογής της μεθοδολογίας TDD. Το TDD προάγει την ποιότητα του κώδικα, καθώς και τη συνεχή βελτίωση μέσα από επαναλαμβανόμενους κύκλους δοκιμών και υλοποίησης.

Η μεθοδολογία **Behavior Driven Development** (BDD) βασίζεται κυρίως στην ιδέα του TDD και αποτελεί μία εξέλιξη της, επηρεασμένη παράλληλα και από διάφορες άλλες agile μεθοδολογίες, η οποία έχει ως βασικό της στόχο την ενθάρρυνση της συνεργασίας μεταξύ όλων των ενδιαφερόμενων μερών στην διαδικασία της ανάπτυξης του λογισμικού ενός έργου. Συγκεκριμένα, όπως αναφέρει και η Fitzgibbons(2021), το BDD προωθεί την τεκμηρίωση και τον σχεδιασμό μίας εφαρμογής με βάσει της συμπεριφοράς που αναμένεται να βιώσει ο χρήστης,

ενθαρρύνοντας τους προγραμματιστές να επικεντρώνονται στις σημαντικές συμπεριφορές που πρέπει να έχει το σύστημα. Έτσι, αποφεύγεται η δημιουργία περιττού κώδικα και υπερβολικών χαρακτηριστικών.

4.2 Η σημασία της συνεργασίας στο BDD

Το BDD επιτυγχάνει τον στόχο της ενθάρρυνσης της συνεργασίας μεταξύ όλων των ενδιαφερόμενων μερών της ανάπτυξης ενός έργου, μέσω της χρήσης μίας κοινής γλώσσας, κατανοητής από όλους, η οποία αναφέρεται ως «**Ομοιογενής Γλώσσα**» (Ubiquitous Language). Χρησιμοποιώντας μία κοινή γλώσσα, βελτιώνεται η κατανόηση των απαιτήσεων του έργου και η ποιότητα του παραγόμενου λογισμικού, αφού όλοι οι εμπλεκόμενοι, ανεξαρτήτως της θέσης τους, μπορούν να συμμετέχουν και να έχουν λόγο στο τελικό αποτέλεσμα.

4.2.1 Ομοιογενής γλώσσα (Ubiquitous Language)

Το κεντρικό σημείο της συνεργασίας μεταξύ όλων των ενδιαφερόμενων μερών είναι η χρήση μίας γλώσσας κατανοητής από όλα τα εμπλεκόμενα δια τμηματικά μέλη των ομάδων. Αυτή η γλώσσα ονομάζεται **Ομοιογενής Γλώσσα** και χρησιμοποιείται σε κάθε επικοινωνία σχετικά με το λογισμικό. Ένα παράδειγμα μίας τέτοιας γλώσσας, που χρησιμοποιείται στην πράξη στο εργαλείο Cucumber και θα γίνει αναλυτική αναφορά παρακάτω, είναι η **Gherkin**. Η Gherkin είναι μία δομημένη γλώσσα με την οποία δημιουργούνται οι έλεγχοι στο BDD και χρησιμοποιεί μία συγκεκριμένη σύνταξη, εύκολη στην ανάγνωση από όλους και γραμμένη σε φυσική γλώσσα (Cucumber, n.d.).

4.2.2 Σενάρια και ιστορίες χρηστών στο BDD

Η πρώτη διαδικασία με την οποία αρχίζει κάθε έργο που αναπτύσσεται με την μεθοδολογία BDD, είναι ο ορισμός της συμπεριφοράς του συστήματος από την οπτική των τελικών χρηστών. Αυτό περιλαμβάνει την δημιουργία **ιστοριών χρηστών** και **σεναρίων** που περιγράφουν πώς θα πρέπει να συμπεριφέρεται το σύστημα σε διάφορες καταστάσεις που μπορεί να έρθει αντιμέτωπο κατά την λειτουργία του.

Οι **ιστορίες χρηστών** γράφονται σε φυσική γλώσσα και είναι της μορφής «Ως [ρόλος του χρήστη], Θέλω να [κάποια ενέργεια του χρήστη με το σύστημα], Ώστε να [κάποιο τελικό αποτέλεσμα, στόχος ή πλεονέκτημα]». Ένα παράδειγμα είναι αυτό της **Εικόνας 8**, όπου παρουσιάζεται μία υποθετική ιστορία χρήστη σε ένα σύστημα βιβλιοθήκης, όπου ο χρήστης ή βιβλιοθηκάριος στην

συγκεκριμένη περίπτωση, θέλει μέσω του συστήματος να μπορεί να χειρίζεται αποδοτικά την διαδικασία δανεισμού ώστε οι δανειζόμενοι να μπορούν να δανείζονται εύκολα βιβλία από την βιβλιοθήκη. (Cucumber, n.d.)

Εικόνα 8. Παράδειγμα ιστορίας χρήστη

```
User Story: As a librarian,
I want to efficiently manage the loaning process of books to registered borrowers
So that borrowers can easily borrow books from the library.
```

Πολλά **σενάρια** μπορούν να ανήκουν σε ένα κοινό χαρακτηριστικό του συστήματος, το ονομαζόμενο «**feature**». Για παράδειγμα, για το χαρακτηριστικό/feature "Εγγραφή χρήστη", μπορούν να υπάρξουν σενάρια που εξετάζουν τι γίνεται στην περίπτωση της επιτυχούς εγγραφής, στην περίπτωση που ο χρήστης έχει ήδη λογαριασμό και πολλά άλλα.

Τα σενάρια γράφονται με την ομοιογενή γλώσσα **Gherkin**, όπως αναφέρθηκε παραπάνω, η οποία έχει την μορφή «**Given-When-Then**»(συνήθως ακολουθείται αυτή η σειρά) και στην οποία θα γίνει αναφορά παρακάτω.

Τα σενάρια που γράφονται με αυτή τη δομή μπορούν εύκολα να είναι κατανοητά από όλους τους ενδιαφερόμενους και βοηθούν στην εξασφάλιση ότι οι απαιτήσεις των χρηστών ικανοποιούνται πλήρως από το σύστημα (Cucumber, n.d.).

4.3 Οι τρείς πρακτικές του BDD

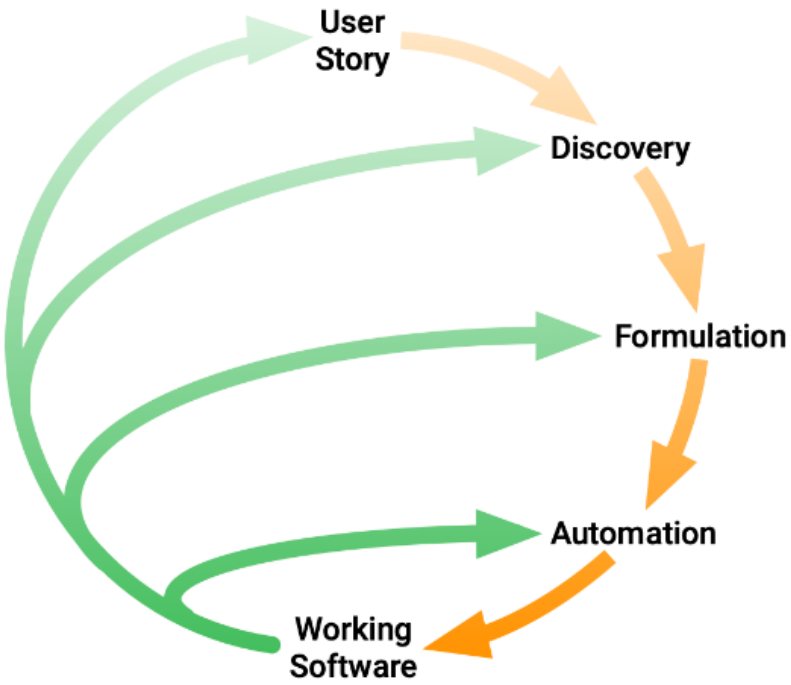
Όπως αναφέρεται και στο documentation της επίσημης σελίδας του Cucumber, το BDD, όπως και το TDD, ενθαρρύνει την εργασία σε γρήγορες επαναλήψεις(rapid iterations), διασπώντας τα προβλήματα των χρηστών σε μικρά κομμάτια ώστε να μπορούν να επιλυθούν γρήγορα και ορθά.

Ουσιαστικά, η καθημερινότητα ενός έργου που αναπτύσσεται με BDD αποτελείται από τρείς διαδικασίες:

1. **Ανάλυση παραδειγμάτων** : Ξεκινά με μια μικρή επερχόμενη αλλαγή, μια ιστορία χρήστη, όπως αναλύθηκε και πριν, και την αναλύει σε πραγματικά παραδείγματα (σενάρια).
2. **Αυτοματοποίηση παραδειγμάτων** : Τροποποιεί τα παραδείγματα ώστε να μπορούν να αυτοματοποιηθούν ως σενάρια Gherkin.

3. **Υλοποίηση κώδικα** : Υλοποιεί τον κώδικα του συστήματος που περιγράφεται από αυτά τα παραδείγματα.

Εικόνα 9. Οι τρείς πρακτικές του BDD



Σημείωση: Ανάκτηση από «Behavior-Driven Development», Cucumber (σύνδεσμος)

4.4 Τεχνικές Ανάλυσης Συνεργασίας

Ένα κομβικό σημείο στην εύρυθμη λειτουργία της μεθοδολογίας BDD, είναι μία συνάντηση η οποία ονομάζεται «Three Amigos Meeting», όπου όπως θα αναλυθεί και παρακάτω, αποτελεί μία «συζήτηση» μεταξύ των διαφορετικών ενδιαφερομένων για την κατανόηση και δημιουργία των απαιτήσεων ή σεναρίων.

Η συνάντηση των τριών φίλων, ή όπως είναι γνωστή «**Three Amigos Meeting**», έχει ως βασικό σκοπό της την μετατροπή των ιστοριών χρηστών σε κατανοητά και ολοκληρωμένα από όλους σενάρια Gherkin. Περιέχει τουλάχιστον τρία διαφορετικά μέλη :

1. **Ιδιοκτήτης του προϊόντος (Product Owner)** - Εστιάζει στο εύρος της εφαρμογής και μετατρέπει τις ιστορίες χρηστών σε features , αποφασίζοντας τι είναι εντός και εκτός του εύρους.

2. **Δοκιμαστής (Tester)** - Δημιουργεί σενάρια και ακραίες περιπτώσεις, εξετάζοντας ποιες ιστορίες χρηστών δεν έχουν καλυφθεί.
3. **Προγραμματιστής (Developer)** - Προσθέτει βήματα στα σενάρια και εξετάζει τις λεπτομέρειες της εφαρμογής, όπως την εκτέλεση και τα πιθανά εμπόδια κατά την δημιουργία του κώδικα.

Οι παραπάνω συζητήσεις δημιουργούν πολύ καλούς ελέγχους, διότι ο το κάθε άτομο βλέπει το σύστημα από διαφορετική οπτική γωνία. Για τον λόγο αυτό, κρίνεται αναγκαίο όλα τα παραπάνω άτομα να παίρνουν μέρος στην συνάντηση, με σκοπό την ανακάλυψη νέων παραδειγμάτων χρήσης. (Cucumber, n.d.)

Εικόνα 10. The Three Amigos Meeting

Σημείωση: Ανακτήθηκε από «The 3 Amigos Meeting», AWH, 2020, Medium. (σύνδεσμος)

4.5 Αυτοματισμός Δοκιμών με το Cucumber

4.5.1 Εισαγωγή στο Cucumber

Το **Cucumber** είναι ένα εργαλείο λογισμικού το οποίο υποστηρίζει την μεθοδολογία του Behavior Driven Development βασιζόμενο στην χρήση της γλώσσας **Gherkin**, για την οποία έγινε μία

μικρή εισαγωγή παραπάνω. Η Gherkin επιτρέπει την περιγραφή των απαιτήσεων του συστήματος και την αυτοματοποίηση των ελέγχων. Συγκεκριμένα, το Cucumber έχει έναν βοηθητικό ρόλο ως προς τον αυτοματισμό των ελέγχων ενός συστήματος, προσφέροντας μία κοινή γλώσσα που είναι κατανοητή από όλους τους ενδιαφερόμενους.

4.5.2 Η σύνταξη Gherkin

Η Gherkin είναι μία δομημένη γλώσσα που χρησιμοποιείται από το εργαλείο Cucumber για τη δημιουργία ελέγχων στο BDD. Η σύνταξή της είναι απλή και κατανοητή, επιτρέποντας σε τεχνικούς και μη τεχνικούς χρήστες να μπορούν να γράφουν σενάρια.

Η γλώσσα Gherkin χρησιμοποιεί την μορφή «Given-Then-When» και με αυτό τον τρόπο επιτρέπει την εύκολη γραφή σεναρίων κατανοητών από όλα τα μέλη.

Οι λέξεις αυτές κλειδιά αποτελούν τον βασικό τρόπο γραφής των σεναρίων ενός feature και κάθε μία από αυτές έχει μία διαφορετική σημασία, όπως αναφέρεται και από το documentation της επίσημης σελίδας του Cucumber :

1. Η λέξη κλειδί **Given**, ορίζει το γενικό πλαίσιο του σεναρίου και βάζει το σύστημα σε μία συγκεκριμένη κατάσταση ώστε να ξεδιπλωθεί το σενάριο στα επόμενα βήματα. Αυτό μπορεί να περιλαμβάνει την προετοιμασία του συστήματος, τη δημιουργία δεδομένων ή την εξασφάλιση ότι το σύστημα βρίσκεται σε μια συγκεκριμένη αρχική κατάσταση.
2. Η λέξη κλειδί **When**, είναι μια ενέργεια, η οποία μεταβάλλει το σύστημα από την κατάσταση «ηρεμίας» του και το αναγκάζει να προβεί σε κάποιο αποτέλεσμα. Αυτή η ενέργεια είναι συνήθως η κεντρική πράξη του σεναρίου και αποτελεί το γεγονός που θα προκαλέσει την αντίδραση του συστήματος.
3. Η λέξη κλειδί **Then**, είναι το αποτέλεσμα, δηλαδή η ενέργεια που αναμένουμε να κάνει το σύστημα όταν γίνεται η ενέργεια στο βήμα When. Αυτό το τμήμα του σεναρίου περιγράφει την αναμενόμενη συμπεριφορά του συστήματος και επιτρέπει την επαλήθευση ότι το σύστημα λειτουργεί όπως πρέπει.

Στην **Εικόνα 11** παρακάτω, παρουσιάζεται ένα παράδειγμα ενός σεναρίου γραμμένου σε Gherkin στο Cucumber, το οποίο αφορά την ενημέρωση των στοιχείων ενός δανειζόμενου σε ένα υποθετικό σύστημα βιβλιοθήκης και το οποίο ανήκει σε ένα feature το οποίο αφορά την

«Διαχείριση Δανειζόμενων». Όπως φαίνεται, κάθε σενάριο περιέχει επίσης έναν σύντομο τίτλο ο οποίος περιγράφει την περίπτωση χρήσης που πραγματεύεται το σενάριο που ακολουθεί.

Εικόνα 11. Παράδειγμα σεναρίου με την μορφή Given-When-Then

```
Scenario: Updating the borrower's details when he is registered
This scenario describes the process of updating the details of a borrower who has already registered before
  Given George Red is registered as a borrower
  When George Red updates his borrowing details
  Then the system saves the changes
```

Στην **Εικόνα 12** παρουσιάζεται ένα ακόμη παράδειγμα ενός υποθετικού σεναρίου στο ίδιο σύστημα με την Εικόνα 11, το οποίο αφορά την περίπτωση της επιτυχούς εγγραφής ενός δανειζόμενου σε ένα σύστημα βιβλιοθήκης.

Εικόνα 12. Παράδειγμα σεναρίου με την μορφή Given-When-Then

```
Scenario: Registering a new borrower
This scenario describes the process of registering a new borrower in the library system
  Given George Red is not registered as a borrower
  When George Red gets registered in the system with a unique borrower number and his details
  Then the system successfully stores the borrower's details
```

4.5.3 Δημιουργία αρχείων χαρακτηριστικών (Feature Files)

Τα **αρχεία χαρακτηριστικών** (feature files), όπως αναφέρθηκε και παραπάνω, είναι αρχεία που περιέχουν τα σενάρια Gherkin που περιγράφουν τις διάφορες περιπτώσεις χρήσης ενός συγκεκριμένου feature του συστήματος. Η δομή των feature files φαίνεται στην **Εικόνα 13** και αποτελείται από την λέξη κλειδί «feature» που αρχικοποιεί το αρχείο. Στην συνέχεια, ακολουθεί το όνομα του feature, το οποίο θα είναι και το όνομα του αρχείου και μία μικρή περιγραφή του feature και των σεναρίων που θα ακολουθήσουν. Ακόμη, μία καλή πρακτική αποτελεί η προσθήκη μίας ιστορία χρήστη, για να γίνει καλύτερα κατανοητό το σύνολο των σεναρίων που θα ακολουθήσουν.

Εικόνα 13. Παράδειγμα δομής ενός feature file

```
Feature: Borrower handling by the system
  The system can register a new person, modify their credentials or delete their account

  User Story: As a librarian,
  I want to efficiently manage the loaning process of books to registered borrowers
  So that borrowers can easily borrow books from the library.

  Scenario: Registering a new borrower
  This scenario describes the process of registering a new borrower in the library system
    Given George Red is not registered as a borrower
    When George Red gets registered in the system with a unique borrower number and his details
    Then the system successfully stores the borrower's details

  Scenario: Borrower trying to register has registered before
  This scenario describes what happens when the system tries to register a new borrower who has already registered before
    Given George Red is registered as a borrower
    When the system attempts to register George Red with a unique borrower number and his details
    Then the system informs that the user already exists
```

Μια καλή πρακτική είναι τα feature files να περιέχουν το πολύ δέκα σενάρια και να εστιάζουν σε ένα συγκεκριμένο χαρακτηριστικό.

4.5.4 Ανάπτυξη των Step Definitions

Τα **Step Definitions** είναι τμήματα κώδικα που συνδέουν τα σενάρια γραμμένα σε Gherkin με την πραγματική υλοποίηση κώδικα ώστε να περάσουν οι έλεγχοι και να δοκιμαστεί το σύστημα. Τα Step Definitions μπορούν να γραφτούν σε διάφορες γλώσσες προγραμματισμού όπως Java, Ruby κ.ά., και προσδιορίζουν την συμπεριφορά κάθε βήματος ενός σεναρίου Gherkin.

Το Cucumber εκτελεί τα σενάρια που βρίσκονται στα feature files τρέχοντας τον αντίστοιχο κώδικα των Step Definitions, διασφαλίζοντας ότι τα βήματα εκτελούνται με σωστή σειρά, προσομοιάζοντας την συμπεριφορά που περιγράφεται στα σενάρια Gherkin.

Η δομή των Step Definitions, όπως φαίνεται και στην **Εικόνα 14**, περιλαμβάνει βοηθητικές δομές δεδομένων και αντικείμενα που χρησιμοποιούνται στον κώδικα των Step Definitions. Για να συνδέσουμε τα βήματα ενός σεναρίου με ένα Step Definition, χρησιμοποιούμε τη μορφή @Given, @When, ή @Then ακολουθούμενη από το κείμενο του βήματος στο σενάριο Gherkin.

Εικόνα 14. Παράδειγμα δομής ενός αρχείου Step Definitions

```
public class StepDefinitions {
    2 usages
    private Person sean;
    3 usages
    private Person lucy;
    2 usages
    private String messageFromSean;

    @Given("Lucy is {int} metres from Sean")
    public void lucy_is_located_metres_from_Sean(Integer distance){
        lucy = new Person();
        sean = new Person();
        lucy.moveTo(distance);
    }
    @When("Sean shouts {string}")
    public void sean_shouts(String message) {
        sean.shout(message);
        messageFromSean = message;
    }
    @Then("Lucy should hear Sean's message")
    public void lucy_should_hear_sean_s_message() {
        assertEquals(asList(messageFromSean), lucy.getMessagesHeard());
    }
}
```

Στην περίπτωση της **Εικόνας 14,** το πρώτο βήμα του Step Definition αρχικοποιεί το περιβάλλον και τα αντικείμενα που θα χρησιμοποιηθούν στα παρακάτω βήματα του σεναρίου, σε ένα υποθετικό σύστημα μίας εφαρμογής που παραδίδει μηνύματα σε άτομα μίας συγκεκριμένης απόστασης.

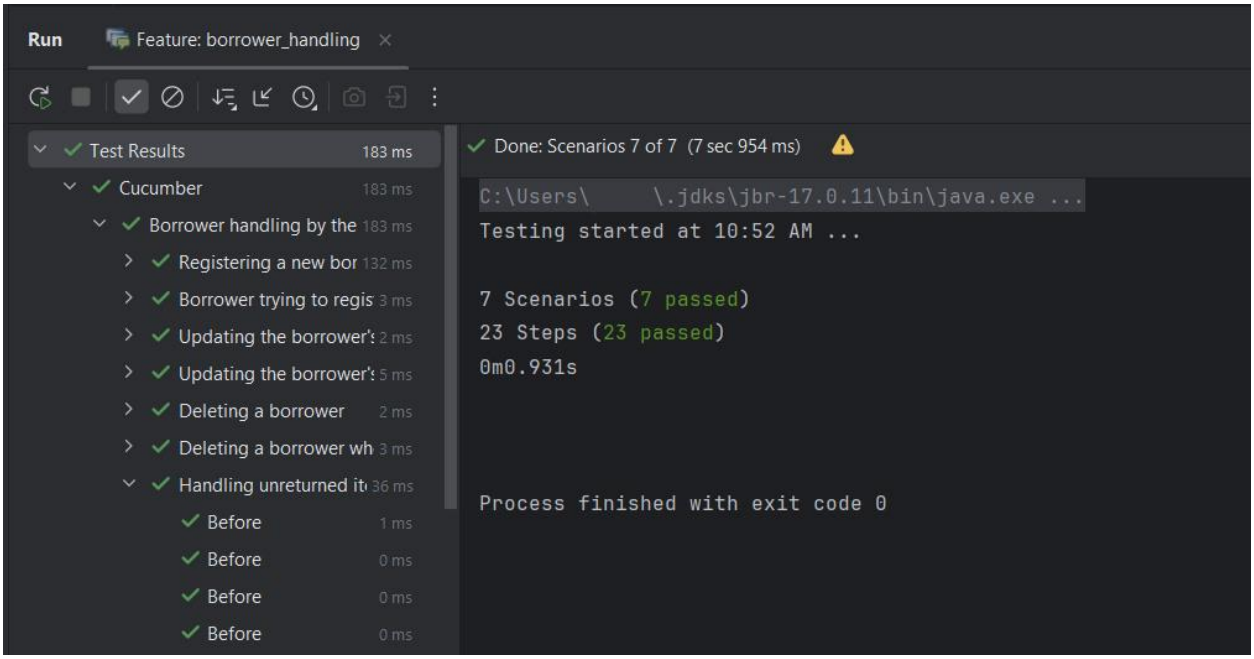
Τα Step Definitions πρέπει να είναι μοναδικά ανεξαρτήτως feature file ώστε κάθε βήμα ενός σεναρίου να μπορεί να αντιστοιχιστεί με ένα μόνο Step Definition, αποφεύγοντας την σύγχυση και διασφαλίζοντας την σωστή εκτέλεση των ελέγχων.

4.6 Αναφορές του Cucumber

Μετά την εκτέλεση των βημάτων των σεναρίων των Step Definitions, το Cucumber παράγει **λεπτομερείς αναφορές** και σχόλια για την εκτέλεση που ακολούθησε. Αυτές οι αναφορές περιλαμβάνουν πληροφορίες για τα βήματα που πέρασαν επιτυχώς, τα βήματα που απέτυχαν και εκείνα που παραλήφθηκαν λόγω κάποιου σφάλματος. Οι αναφορές αυτές αποτελούν σημαντικό

εργαλείο για προγραμματιστές και τους ειδικούς ελέγχου, καθώς τους βοηθούν να κατανοήσουν την κατάσταση του συστήματος και να εντοπίσουν τυχόν προβλήματα. (Cucumber, n.d.)

Εικόνα 15. Παράδειγμα αναφοράς με το Cucumber



4.7 Ζωντανή τεκμηρίωση

Όταν χρησιμοποιείται το BDD ως μεθοδολογία ανάπτυξης ενός συστήματος, γράφονται παραδείγματα ή σενάρια τα οποία καθοδηγούν την ανάπτυξη. Αυτά τα παραδείγματα χρησιμεύουν επίσης ως κριτήρια αποδοχής (acceptance criteria) και είναι γραμμένα σε μια κοινή γλώσσα, ώστε να είναι κατανοητά από όλους τους εμπλεκόμενους. Επομένως, τα παραδείγματα ή σενάρια αυτά λειτουργούν και ως μία «**ζωντανή» τεκμηρίωση** του συστήματος. Σε αντίθεση με τα παραδοσιακά έγγραφα κριτηρίων αποδοχής, αυτή η τεκμηρίωση ενημερώνεται συνεχώς με κάθε νέα αλλαγή ή προσθήκη λειτουργικότητας στο σύστημα.

Η σημασία της ζωντανής τεκμηρίωσης είναι ιδιαίτερα σημαντική σε ένα περιβάλλον ανάπτυξης λογισμικού, καθώς επιτρέπει σε όλα τα μέλη της ομάδας να κατανοούν τις απαιτήσεις του συστήματος. Αυτό είναι ιδιαίτερα χρήσιμο για νέα μέλη της ομάδας, που δεν γνωρίζουν από πριν το σύστημα που αναπτύσσεται. Η ζωντανή τεκμηρίωση διασφαλίζει ότι η γνώση του συστήματος διατηρείται και διαμοιράζεται αποτελεσματικά σε όλη την ομάδα (Cucumber, n.d.).

4.8 Καλές πρακτικές για γραφή με Cucumber

Η χρήση του Cucumber για τον αυτοματισμό των ελέγχων και την υιοθέτηση της μεθοδολογίας BDD μπορεί να συμβάλλει σημαντικά στην βελτίωση της ποιότητας του παραγόμενου λογισμικού. Ωστόσο, η επίτευξη αυτού του στόχου εξαρτάται σε μεγάλο βαθμό από τον τρόπο που θα γίνει η γραφή των σεναρίων και η οργάνωσή τους σε feature files. Μερικές καλές πρακτικές για την συγγραφή σεναρίων με Cucumber, όπως αναφέρει και ο Pathak (2022), είναι οι εξής:

1. **Μικρά σενάρια** : Τα σενάρια που γράφονται για ένα feature και τα οποία περιγράφουν μία περίπτωση χρήσης του συστήματος, θα πρέπει να είναι μικρά και να εστιάζουν σε μία μόνο λειτουργία του συστήματος.
2. **Χρήση παραδειγμάτων από τον πραγματικό κόσμο** : Τα σενάρια θα πρέπει να βασίζονται σε πραγματικά παραδείγματα ώστε να διασφαλίζουν ότι είναι ρεαλιστικά και ότι το σύστημα θα λειτουργήσει σωστά σε πραγματικές συνθήκες
3. **Επαναχρησιμοποίηση βημάτων** : Όταν είναι δυνατόν, θα πρέπει να γίνεται επαναχρησιμοποίηση των βημάτων των σεναρίων ώστε να μειωθεί η επανάληψη και να διευκολυνθεί η συντήρηση των σεναρίων.
4. **Οργάνωση των feature files** : Τα feature files θα πρέπει να οργανώνονται με τρόπο που να διευκολύνει την εύρεση και κατανόηση των σεναρίων. Κάθε αρχείο χαρακτηριστικών θα πρέπει να περιέχει σενάρια που σχετίζονται με ένα συγκεκριμένο χαρακτηριστικό ή λειτουργία του συστήματος, χρησιμοποιώντας κατανοητούς τίτλους και περιγραφές.

5. Χρήση Μεγάλων Γλωσσικών Μοντέλων για BDD

Το παρόν κεφάλαιο αναλύει την μεθοδολογία, τα κριτήρια αξιολόγησης αλλά και την εμπειρική αξιολόγηση του πειράματος που πραγματοποιήθηκε στο πλαίσιο της εργασίας.

5.1 Μεθοδολογία

Στην παρούσα εργασία, ακολουθήθηκε μια συστηματική προσέγγιση για την αξιολόγηση της απόδοσης διαφορετικών μεγάλων γλωσσικών μοντέλων σε ένα συγκεκριμένο πλαίσιο χρήσης, βασισμένο στο πρόβλημα της διαχείρισης βιβλιοθήκης.

5.1.1 Το πρόβλημα της βιβλιοθήκης

Ως αρχική αναφορά για το πείραμα που πραγματεύεται η εργασία, επιλέχθηκε το πρόβλημα της διαχείρισης μίας βιβλιοθήκης. Το πρόβλημα αυτό της διαχείρισης της βιβλιοθήκης, αποτελεί μια εφαρμογή η οποία αποτελείται από μία σειρά τυπικών use cases (περιπτώσεων χρήσης) που καλύπτουν τις βασικές λειτουργίες μίας βιβλιοθήκης, όπως για παράδειγμα:

1. **Η αναζήτηση αντιγράφων βιβλίων** : Εύρεση αντιγράφων βιβλίων στην συλλογή της βιβλιοθήκης με βάση διάφορα κριτήρια.
2. **Η επιστροφή αντιγράφων βιβλίων** : Διαχείριση της διαδικασίας επιστροφής βιβλίων από τους δανειζόμενους και λειτουργίες σε περίπτωση καθυστέρησης.
3. **Η διαχείριση δανειζόμενων** : Εγγραφή νέου δανειζόμενου, ενημέρωση των στοιχείων του, διαγραφή των στοιχείων του.

Το συγκεκριμένο πρόβλημα επιλέχθηκε ως αναφορά για την εργασία διότι αποτελεί την βάση του μαθήματος της «Τεχνολογίας Λογισμικού» και έχει δημιουργηθεί από τον επιβλέποντα δάσκαλο της εργασίας, τον κύριο Διαμαντίδη, και με το οποίο μπορούν να δημιουργηθούν με ευκολία σενάρια δοκιμών και αυτοματοποιημένος κώδικας με χρήση της μεθοδολογίας BDD και του εργαλείου Cucumber, για να γίνει στην συνέχεια η σύγκριση με τα αποτελέσματα των μεγάλων γλωσσικών μοντέλων (σύνδεσμος του συστήματος της βιβλιοθήκης στο GitHub).

5.1.2 Μετατροπή των use cases σε σενάρια Cucumber

Αφού επιλέχθηκε το πρόβλημα της διαχείρισης της βιβλιοθήκης ως αναφορά, το επόμενο βήμα για την βαθιά κατανόηση του συστήματος της βιβλιοθήκης αλλά και της μεθοδολογίας του BDD και της χρήσης του εργαλείου Cucumber, ήταν η μετατροπή των use cases της βιβλιοθήκης σε

σενάρια γραμμένα με την γλώσσα Gherkin. Αυτή η διαδικασία επέτρεψε την αυτοματοποίηση των σεναρίων με το εργαλείο Cucumber και την δημιουργία των BDD tests, παρέχοντας ένα μέτρο σύγκρισης για τα αποτελέσματα των γλωσσικών μοντέλων. Για κάθε use case του συστήματος της βιβλιοθήκης, το οποίο ήταν εντός του εύρους της εφαρμογής, δημιουργήθηκε το αντίστοιχο feature αποτελούμενο από τα αντίστοιχα σενάρια σε Gherkin, που περιγράφουν την συμπεριφορά του συστήματος με βάση πραγματικά παραδείγματα. Στην **Εικόνα 16** παρουσιάζεται ένα παράδειγμα ενός use case του συστήματος της βιβλιοθήκης, που περιγράφει την βασική ροή της διαδικασίας του δανεισμού ενός αντιτύπου σε έναν δανειζόμενο αλλά και όλες τις εναλλακτικές ροές που αυτό περιέχει.

Εικόνα 16. Παράδειγμα του use case δανεισμού αντιτύπων του συστήματος της βιβλιοθήκης

βασική ροή «δανεισμός αντιτύπων»

1. Ο δανειζόμενος έρχεται στο βιβλιοθηκονόμο κρατώντας τα αντίτυπα των βιβλίων προς δανεισμό.
2. Ο βιβλιοθηκονόμος αναζητά τον δανειζόμενο.
3. Το Σύστημα παρουσιάζει τα στοιχεία του δανειζομένου.
4. Ο βιβλιοθηκονόμος αναζητά το αντίτυπο.
5. Το Σύστημα παρουσιάζει τα στοιχεία του αντιτύπου.
6. Ο βιβλιοθηκονόμος επιλέγει το αντίτυπο προς δανεισμό.
7. Το Σύστημα επιβεβαιώνει ότι ο δανειζόμενος μπορεί να δανειστεί το αντίτυπο.
8. Το Σύστημα καταχωρίζει το δανεισμό και εμφανίζει την προθεσμία επιστροφής.
9. Ο βιβλιοθηκονόμος ενημερώνει τον δανειζόμενο για την προθεσμία επιστροφής του αντιτύπου.

Ο βιβλιοθηκονόμος επαναλαμβάνει τα βήματα 4 έως 9 για όλα τα αντίτυπα.

εναλλακτικές ροές «δανεισμός αντιτύπων»

* Σε οποιοδήποτε σημείο το λογισμικό καταρρέει.
  1. Ο βιβλιοθηκονόμος εκκινεί το Σύστημα.
  2. Το Σύστημα ταυτοποιεί το βιβλιοθηκονόμο.
  3. Ο βιβλιοθηκονόμος εκκινεί το δανεισμό για τα εναπομείναντα αντίτυπα.

2α. Ο δανειζόμενος έρχεται για πρώτη φορά για δανεισμό.
  1. Ο βιβλιοθηκονόμος επιβεβαιώνει ότι ο δανειζόμενος μπορεί να δανειστεί βιβλία από τη Βιβλιοθήκη.
    1α. Ο δανειζόμενος δε δικαιούται να δανειστεί από τη Βιβλιοθήκη.
      1. Ο δανεισμός τερματίζει.
  2. Ο βιβλιοθηκονόμος καταχωρίζει τον δανειζόμενο στο σύστημα με τη Διαχείριση Δανειζομένου.

εναλλακτικές ροές «δανεισμός αντιτύπων»

5α. Το Σύστημα δε βρίσκει το αντίτυπο του βιβλίου
  1. Ο βιβλιοθηκονόμος κρατά το αντίτυπο για να διαπιστώσει το σφάλμα αργότερα.
  2. Ο δανεισμός τερματίζει.

7α. Ο δανειζόμενος δεν μπορεί να δανειστεί βιβλία.
  1. Ο βιβλιοθηκονόμος ενημερώνει το δανειζόμενο.
  2. Κρατά τα εναπομείναντα αντίτυπα για να επιστρέψουν στα ράφια.
  3. Ο δανεισμός τερματίζει.

Σημείωση: Ανάκτηση από «Τεχνολογία Λογισμικού Β'ΕΚΔΟΣΗ», των Ε.Α.Γιακουμάκη και Ν.Α. Διαμαντίδη, σελ. 127-128 (2021)

Στην συνέχεια, το συγκεκριμένο use case μετατράπηκε σε feature με σενάρια που περιγράφουν τις περιπτώσεις χρήσης. Όπως φαίνεται και στην **Εικόνα 17**, το feature «Loaning Items» αποτελείται από διάφορα σενάρια που χρησιμοποιούν χαρακτήρες ή «περσόνες» όπως αναφέρονται σε πολλά άρθρα, για την αληθοφανή παρουσίαση του παραδείγματος και της καλύτερης κατανόησης από τους αναγνώστες.

Εικόνα 17. Παράδειγμα σεναρίων Cucumber για το use case του δανεισμού αντιτύπων

```
Feature: Loaning items
  The library application allows for the librarian to loan an item of a book to a borrower based
  on some conditions of the system

  User story: As a library member
  I want to be able to borrow items
  So that I can study them at home


  Scenario: Successful loaning of an item
  This scenario describes the successful process of loaning an item to a borrower that is entitled to borrow
    Given the library has the item Harry Potter available
    And George Red is a registered borrower
    And George Red has 2 pending items to be returned
    And George Red has been assigned a maximum lending limit of 5
    When George Red borrows the item Harry Potter
    Then the system successfully loans the item Harry Potter to George Red with a due date set
    And George Red's pending items increase to 3

  Scenario: Borrower can borrow only one item due to his lending limit
  This scenario describes the successful process of loaning only one but not two items to a borrower that is entitled to borrow only one item
    Given the library has the items Harry Potter and Moby Dick available
    And George Red is a registered borrower
    And George Red has 2 pending items to be returned
    And George Red has been assigned a maximum lending limit of 3
    When George Red tries to borrow both items
    Then the system successfully loans the item Harry Potter to George Red with a due date set
    And the system does not loan Moby Dick to George Red due to the lending limit reached
    And George Red's pending items increase to 3

  Scenario:Item not found
  This scenario describes the edge case where the library system cannot find the item, so the loan isn't happening
   This scenario describes the edge case where the library system cannot find the item, so the loan isn't happening
     Given the item Harry Potter is in the library but not in the system
     And George Red is a registered borrower
     When George Red tries to borrow the item Harry Potter
     Then the system returns an error due to the item's status
     And the system withdraws the item Harry Potter

   Scenario:The borrower is not eligible to borrow
   This scenario describes the unsuccessful process of loaning an item to a borrower that has reached his max lending limit
     Given the library has the item Harry Potter available
     And George Red is a registered borrower
     And George Red has 3 pending items to be returned
     And George Red has been assigned a maximum lending limit of 3
     When George Red tries to borrow the item Harry Potter
     Then the system doesn't allow the loan
     And George Red's pending items remain 3
```

Συγκεκριμένα, δημιουργήθηκαν σενάρια για την περίπτωση του επιτυχή δανεισμού των αντιτύπων, για την περίπτωση της ύπαρξης ορίου για τον δανεισμό από τον δανειζόμενο, για την μη-ύπαρξη ενός αντιτύπου αλλά και σενάρια για την περίπτωση που ο δανειζόμενος δεν πληρεί τα κριτήρια δανεισμού.

5.1.3 Ανάπτυξη BDD tests

Η επόμενη φάση μετά την συγγραφή των σεναρίων Cucumber για τα use cases του προβλήματος της βιβλιοθήκης, ήταν η δημιουργία των αυτοματοποιημένων τεστ, γνωστών και ως Step Definitions, τα οποία χρησιμοποιήθηκαν από το Cucumber για την επαλήθευση των σεναρίων.

Για κάθε σενάριο δημιουργήθηκαν αντίστοιχες κλάσεις και μέθοδοι στην γλώσσα προγραμματισμού Java, που έλεγχαν την λειτουργία του συγκεκριμένου βήματος του σεναρίου με το πραγματικό σύστημα βιβλιοθήκης.

Κατά την συγγραφή του κώδικα των Step Definitions, έγινε χρήση αντικειμένων Data Access Objects με σκοπό την προσομοίωση μίας πραγματικής βάσης δεδομένων, αποθηκεύοντας δεδομένα και μεταβλητές προερχόμενα από τα τεστ. Επίσης, χρησιμοποιήθηκε η τεχνική «Dependency Injection» για τη μείωση της πολυπλοκότητας του κώδικα και την αύξηση της επαναχρησιμοποίησής του, η οποία ουσιαστικά δημιουργεί μία κοινή κλάση όπου περιέχει όλες τις βοηθητικές μεθόδους για να χρησιμοποιεί κάθε διαφορετικό feature file. Ωστόσο, η τεχνική αυτή δεν ζητήθηκε από τα μεγάλα γλωσσικά μοντέλα, καθώς αύξανε την πολυπλοκότητα χωρίς να προσφέρει πρόσθετη πληροφορία. Στην **Εικόνα 18** φαίνεται ο κώδικας των Step Definitions που γράφτηκε για το feature «Loaning Items» όπου αναφέρθηκε παραπάνω.

Εικόνα 18. Παράδειγμα κώδικα σε Java για τον έλεγχο των σεναρίων

```java
package hellocucumber.StepDefinitions;

import ...

public class itemLoaningStepDefinitions {  Konstantinos Platias
    // Instance variables
    private LocalDate dueDate1; //Due date for the book Harry Potter  5 usages
    private LocalDate dueDate2; //Due date for the book Moby Dick  2 usages
    private Borrower georgeRed;  12 usages
    private Item hPotterItem;  10 usages
    private Item mDickItem;  2 usages
    private final LibraryWorld world;  18 usages
    public itemLoaningStepDefinitions(LibraryWorld world){  no usages  Konstantinos Platias
        this.world=world;
    }
    @Before  Konstantinos Platias
    public void setUp(){
        world.clearItems();
        world.clearLoans();
        world.clearBorrowers();
    }
    @Given("the library has the item Harry Potter available")  Konstantinos Platias
    public void givenItemHarryPotterAvailable() {
        // Create and save the item/book Harry Potter
        hPotterItem=world.createItem( title: "Harry Potter",  number: 1001112);
    }
    @Given("{borrower} is a registered borrower")  Konstantinos Platias
    public void givenBorrowerRegistered(Borrower borrower) {
        georgeRed = borrower;
    }
    @Given("George Red has {int} pending items to be returned")  Konstantinos Platias
    public void givenPendingItemsToBeReturned(Integer pendingItems) {
        // Adding dummy loans to create pending items (not reflected in DAO)
        for(int i=0;i<pendingItems;i++) {
            Loan dummy_loan = new Loan();
            dummy_loan.setBorrower(georgeRed);
        }
    }
    @Given("George Red has been assigned a maximum lending limit of {int}")  Konstantinos Platias
    public void givenBorrowerMaxLendingLimit(Integer maxLendingLimit) {
        BorrowerCategory category = new BorrowerCategory();
        category.setMaxLendingItems(maxLendingLimit);
        category.setMaxLendingDays(5); //dummy number that will be used later
        georgeRed.setCategory(category);
    }
    @When("George Red borrows the item Harry Potter")  Konstantinos Platias
    public void  whenBorrowerBorrowsItemHarryPotter() {
        // Attempt to borrow the item and get due date if successful
        if(world.loanService.findBorrower(georgeRed.getBorrowerNo())){
            dueDate1 = world.loanService.borrow(hPotterItem.getItemNumber());
        }
    }
    @Then("the system successfully loans the item Harry Potter to George Red with a due date set")  Konstantinos Platias
    public void thenSystemLoansItemHarryPotterToGeorgeRed() {
        Assertions.assertEquals(georgeRed,world.loanDao.findPending(hPotterItem.getItemNumber()).getBorrower());
        Assertions.assertNotNull(dueDate1);
    }
    @Then("George Red's pending items increase to {int}")  Konstantinos Platias
    public void thenPendingItemsIncrease(Integer pendingItems) {
        Assertions.assertEquals(pendingItems, georgeRed.countPendingItems());
    }
    @Given("the library has the items Harry Potter and Moby Dick available")  Konstantinos Platias
```

```java
public void givenItemsHarryPotterAndMobyDickAvailable() {
    // Create and save the book and the item Harry Potter
    hPotterItem=world.createItem( title: "Harry Potter",  number: 1001112);
    // Create and save the book and the item Moby Dick
    mDickItem = world.createItem( title: "Moby Dick", number: 1001233);
}

@When("George Red tries to borrow both items")  Konstantinos Platias
public void whenBorrowerTriesToBorrowBothItems() {
    if(world.loanService.findBorrower(georgeRed.getBorrowerNo())){
        dueDate1 = world.loanService.borrow(hPotterItem.getItemNumber());
    }
    if(world.loanService.findBorrower(georgeRed.getBorrowerNo())){
        dueDate2 = world.loanService.borrow(mDickItem.getItemNumber());
    }
}

@Then("the system does not loan Moby Dick to George Red due to the lending limit reached")  Konstantinos Platias
public void thenSystemDoesNotLoanMobyDickToGeorgeRed() {
    Assertions.assertNull(dueDate2);
    Assertions.assertEquals(georgeRed.countPendingItems(), georgeRed.getCategory().getMaxLendingItems());
}

@Given("the item Harry Potter is in the library but not in the system")  Konstantinos Platias
public void givenItemHarryPotterInLibraryNotInSystem() {
    // Create Harry Potter item and mark it as withdrawn
    hPotterItem=world.createItem( title: "Harry Potter",  number: 1001112);
    hPotterItem.withdraw();
}


    @When("George Red tries to borrow the item Harry Potter")  Konstantinos Platias
    public void whenBorrowerTriesToBorrowItemHarryPotter() {
        if(world.loanService.findBorrower(georgeRed.getBorrowerNo())){
            dueDate1 = world.loanService.borrow(hPotterItem.getItemNumber());
        }
    }
    @Then("the system returns an error due to the item's status")  Konstantinos Platias
    public void thenSystemReturnsErrorDueToItemsStatus() { Assertions.assertNull(dueDate1); }
    @Then("the system withdraws the item Harry Potter")  Konstantinos Platias
    public void thenSystemWithdrawsItemHarryPotter() {
        // Assert that the item's state is withdrawn
        Assertions.assertEquals(ItemState.WITHDRAWN, hPotterItem.getState());
    }

    @Then("the system doesn't allow the loan")  Konstantinos Platias
    public void thenSystemDoesNotAllowLoan() {
        Assertions.assertNull(world.loanDao.findPending(hPotterItem.getItemNumber()));
    }
    @Then("George Red's pending items remain {int}")  Konstantinos Platias
    public void thenBorrowerPendingItemsRemain(Integer pendingItems) {
        Assertions.assertEquals(pendingItems, georgeRed.countPendingItems());
    }
}
```

5.1.4 Εκτέλεση και επαλήθευση των BDD Tests

Μετά την ανάπτυξη των BDD tests, τα σενάρια εκτελέστηκαν για να επαληθευτεί η ορθότητα των σεναρίων σε Gherkin. Η εκτέλεση των BDD tests περιλάμβανε την αυτόματη εκτέλεση των σεναρίων από το εργαλείο Cucumber, το οποίο αυτοματοποιεί αυτή την διαδικασία και παράγει αναφορές σχετικά με τα αποτελέσματα των ελέγχων που έγιναν, όπως φαίνεται στην **Εικόνα 15** του προηγουμένου κεφαλαίου.

Τα BDD tests/Step Definitions επαλήθευσαν αν το σύστημα ανταποκρινόταν σωστά στις δεδομένες συνθήκες και αν τα αποτελέσματα συμφωνούσαν με τις αναμενόμενες συμπεριφορές που περιγράφονται στα σενάρια.

Συνοψίζοντας, η χρήση του συστήματος της βιβλιοθήκης, η μετατροπή των use cases σε σενάρια Cucumber και η ανάπτυξη των BDD tests αποτέλεσε ένα κρίσιμο βήμα στη μεθοδολογία της εργασίας. Η διαδικασία αυτή επέτρεψε τη συστηματική αξιολόγηση της ικανότητας των μεγάλων γλωσσικών μοντέλων να εκτελούν τις προβλεπόμενες λειτουργίες της βιβλιοθήκης και έθεσε τις βάσεις για τη σύγκριση των διαφορετικών μοντέλων με αντικειμενικά κριτήρια.

5.1.5 Επιλογή συγκεκριμένων μεγάλων γλωσσικών μοντέλων

Για την υλοποίηση του πειράματος, επιλέχθηκαν συγκεκριμένα μεγάλα γλωσσικά μοντέλα που διακρίνονται για την ταχύτητα και την ικανότητά τους να παράγουν ποιοτικά αποτελέσματα. Η επιλογή αυτών των μεγάλων γλωσσικών μοντέλων κρίθηκε απαραίτητη για την εξαγωγή αξιόπιστων συμπερασμάτων σχετικά με την απόδοσή τους στην παραγωγή αυτοματοποιημένου κώδικα.

Το πρώτο μοντέλο που χρησιμοποιήθηκε ήταν το **GitHub Copilot** της Microsoft, το οποίο βασίζεται στο μοντέλο GPT-3.5 Turbo. Το GitHub Copilot χρησιμοποιείται ειδικά μόνο για την παραγωγή κώδικα και λειτουργεί ως ένας παράλληλος βοηθός για την υποβοήθηση και συμπλήρωση των προγραμματιστών. Η εξειδίκευσή του στην παραγωγή κώδικα το διαφοροποιεί από τα υπόλοιπα μοντέλα που χρησιμοποιήθηκαν στην εργασία, καθιστώντας το ένα εξαιρετικά χρήσιμο εργαλείο

Στην συνέχεια εξετάστηκαν όλα τα μεγάλα γλωσσικά μοντέλα της OpenAI, δηλαδή το **Chat GPT-3.5**, το **Chat GPT-4** και το **Chat GPT-4o (omni).** Τα μοντέλα αυτά κατατάσσονται κατά αύξουσα ισχύ και επίπεδο πρόσβασης, από το λιγότερο ισχυρό στο πιο ισχυρό. Η OpenAI, ως πρωτοπόρος

στη δημιουργία μεγάλων γλωσσικών μοντέλων, έχει αναπτύξει τα πιο δημοφιλή μοντέλα της αγοράς έως σήμερα. Η επιλογή των μοντέλων της OpenAI στην εργασία βασίζεται στη γενική αναγνώριση της και την δύναμη που κατέχουν τα μοντέλα της, κάτι που δικαιολογεί την χρήση όλων τους στην αξιολόγηση.

Συνολικά, η επιλογή των παραπάνω μεγάλων γλωσσικών μοντέλων παρέχει μια αντικειμενική αξιολόγηση της ικανότητάς τους να υποστηρίξουν την ανάπτυξη λογισμικού και συγκεκριμένα το έργο της παραγωγής αυτοματοποιημένου κώδικα με περιορισμένες πληροφορίες.

5.1.6 Σχεδιασμός διαφορετικών συνομιλιών βάσει προϋπάρχουσας γνώσης

Ένα κρίσιμο βήμα στην διεξαγωγή του πειράματος και στην αξιολόγηση του αυτοματοποιημένου κώδικα τεστ που παρήγαγαν τα μεγάλα γλωσσικά μοντέλα που αναφέρθηκαν παραπάνω, ήταν ο σχεδιασμός και η διαμόρφωση διαφορετικών συνομιλιών με βάση την προϋπάρχουσα γνώση που δινόταν στο σύστημα κάθε φορά. Στο πλαίσιο της εργασίας, δημιουργήθηκαν ποικίλες συνομιλίες με κάθε ένα από τα μεγάλα γλωσσικά μοντέλα για να δοκιμαστεί η απόδοσή τους σε διαφορετικά επίπεδα προϋπάρχουσας γνώσης.

Η αξιολόγηση χωρίστηκε σε τέσσερις φάσεις, με κάθε φάση να προσθέτει νέες πληροφορίες στις προηγούμενες, διαφοροποιώντας έτσι την ποσότητα και την ποιότητα της γνώσης που διατίθεται στο μοντέλο σε κάθε στάδιο. Συγκεκριμένα :

1. **Πρώτη φάση (Phase 1)** : Σε αυτή τη φάση, το μεγάλο γλωσσικό μοντέλο έλαβε περιορισμένη προϋπάρχουσα γνώση. Συγκεκριμένα, δόθηκε μια βασική περιγραφή της αρχιτεκτονικής του συστήματος της βιβλιοθήκης, γενικές πληροφορίες για τις απαντήσεις που αναμένονται και όλα τα σενάρια των διάφορων λειτουργιών γραμμένα σε Gherkin μέσω του εργαλείου Cucumber.
2. **Δεύτερη φάση (Phase 2)** : Στη δεύτερη φάση, η γνώση που δόθηκε ήταν η ίδια με την προηγούμενη φάση, με την προσθήκη νέας πληροφορίας, τα ονόματα των Domain κλάσεων του συστήματος της βιβλιοθήκης, που θα χρειαζόταν να χρησιμοποιήσει για την δημιουργία των αυτοματοποιημένων τεστ.
3. **Τρίτη φάση (Phase 3)** : Η τρίτη φάση με την σειρά της περιείχε την ίδια γνώση με την φάση 2, αλλά στο σημείο αυτό εισάχθηκε σαν επιπλέον προϋπάρχουσα γνώση οι ιδιότητες που χρησιμοποιεί κάθε μία από τις Domain κλάσεις που δόθηκαν, ώστε να μπορέσει το μοντέλο να χρησιμοποιήσει τις ιδιότητες αυτές στην παραγωγή των τεστ.

4. **Τέταρτη φάση (Phase 4)** : Στην τέταρτη και τελευταία φάση, επεκτάθηκε η γνώση της τρίτης φάσης με την εισαγωγή των υπογραφών μεθόδων που χρησιμοποιούνται από τις Domain κλάσεις. Αυτή η φάση αντιπροσωπεύει την μέγιστη ποσότητα προϋπάρχουσας γνώσης που δόθηκε στα μεγάλα γλωσσικά μοντέλα, επιτρέποντας την πληρέστερη κατανόηση του συστήματος και τη δημιουργία των πιο σύνθετων και ακριβών αυτοματοποιημένων τεστ.

Για κάθε μία από τις παραπάνω κατηγορίες προϋπάρχουσας γνώσης χρησιμοποιήθηκαν διαφορετικές τεχνικές παρουσίασης της γνώσης αυτής στα μεγάλα γλωσσικά μοντέλα, προκειμένου να βρεθεί ο αποδοτικότερος τρόπος επικοινωνίας με τα μεγάλα γλωσσικά μοντέλα. Σκοπός αυτής της προσέγγισης ήταν η ανάπτυξη μιας **μεθοδολογίας** που θα εξασφάλιζε τη βέλτιστη επικοινωνία με τα μοντέλα για την παραγωγή αυτοματοποιημένου κώδικα τεστ.

Ειδικότερα, κατά τη διάρκεια κάθε συζήτησης με τα γλωσσικά μοντέλα, εφαρμόστηκαν τέσσερις διαφορετικές τεχνικές παρουσίασης της γνώσης, οι οποίες ήταν οι εξής:

1. **Παρουσίαση όλων των σεναρίων Cucumber σε μεμονωμένα μηνύματα ή σε διαφορετικά μηνύματα** : Σε αυτήν την τεχνική, η προϋπάρχουσα γνώση για τα σενάρια Gherkin που σχετίζονταν με το σύστημα δινόταν στο μεγάλο γλωσσικό μοντέλο σε ένα ενιαίο μήνυμα (prompt) ή σε ξεχωριστά μηνύματα. Στη συνέχεια, ζητούνταν από το μοντέλο να δημιουργήσει τον κώδικα για τα Step Definitions που αντιστοιχούσαν για τα συγκεκριμένα σενάρια.
2. **Εντολή στο μεγάλο γλωσσικό μοντέλο να παράγει ή να μην παράγει τον κώδικα των Domain κλάσεων πρώτα** : Σε αυτή την τεχνική, αφού επιλέγονταν ένας από τους παραπάνω τρόπους παρουσίασης, ζητούσαμε επιπλέον, μετά την παρουσίαση των σεναρίων σε Cucumber, τη δημιουργία ή την μη δημιουργία του κώδικα για τις Domain κλάσεις που το μοντέλο θεωρούσε απαραίτητο. Ο στόχος ήταν να εξετάσουμε αν η παραγωγή του κώδικα των Domain κλάσεων πρώτα θα βελτίωνε την ποιότητα των αποτελεσμάτων στην παραγωγή των αυτοματοποιημένων τεστ ή εάν μη παρουσιάζοντας την εντολή αυτή, τα αποτελέσματα ήταν χειρότερα.

Η εφαρμογή αυτών των τεχνικών είχε ως σκοπό την κατανόηση του τρόπου με τον οποίο οι διάφορες μορφές παρουσίασης γνώσης επηρεάζουν την απόδοση των γλωσσικών μοντέλων στην

παραγωγή αυτοματοποιημένου κώδικα, και συνεισέφερε στην ανάπτυξη στρατηγικών για τη βελτίωση της αποδοτικότητας της επικοινωνίας με τα μοντέλα.

5.2 Κριτήρια αξιολόγησης

Η αξιολόγηση της απόδοσης των μεγάλων γλωσσικών μοντέλων στο πλαίσιο της εργασίας στην παραγωγή του αυτοματοποιημένου κώδικα των σεναρίων BDD, βασίζεται σε ένα σύνολο κριτηρίων τα οποία εξασφαλίζουν μία συστηματική και αντικειμενική προσέγγιση στην ανάλυση των αποτελεσμάτων. Τα κριτήρια αυτά εστιάζουν σε διάφορες πτυχές της απόδοσης των μεγάλων γλωσσικών μοντέλων, από την ακρίβεια των απαντήσεων έως μέχρι την αποδοτικότητα της παραγωγής των μηνυμάτων.

Συγκεκριμένα, τα κριτήρια αξιολόγησης περιλαμβάνουν παράγοντες όπως ο αριθμός των μηνυμάτων που απαιτούνται για να ληφθεί μια πλήρης απάντηση, καθώς και το ποσοστό των απαντήσεων που κρίνονται αποδεκτές βάσει των απαιτήσεων του σεναρίου BDD.

Ο **Πίνακας 1** παραθέτει τα κριτήρια αξιολόγησης, με κάθε σειρά να αναφέρεται σε ένα διαφορετικό αριθμημένο κριτήριο. Στις στήλες του πίνακα παρουσιάζονται οι απαραίτητες πληροφορίες για κάθε κριτήριο, όπως το όνομα, η περιγραφή, η μονάδα μέτρησης, καθώς και ο τύπος που χρησιμοποιείται για τον υπολογισμό του αποτελέσματος που περιγράφει το κριτήριο, όπου αυτό είναι εφικτό.

Πίνακας 1. Κριτήρια αξιολόγησης των μεγάλων γλωσσικών μοντέλων

Αριθμός Κριτηρίου	Όνομα Κριτηρίου	Περιγραφή	Μέτρηση	Τύπος
1	Παροχή των features ταυτόχρονα	Εάν παρέχουμε τα features στο LLM όλα μαζί ή σε περισσότερα από ένα μηνύματα.	Ναι/Όχι	-
2	Παροχή κώδικα Domain/DAOs/Services από την αρχή	Εάν δώσαμε οδηγίες στον LLM να παράγει τον κώδικα Domain/DAOs/Services πρώτα ή όχι.	Ναι/Όχι	-
3	Απαιτούμενα μηνύματα για ολοκληρωμένα Step Definitions	Καταμέτρηση του αριθμού των μηνυμάτων που απαιτούνται για τη δημιουργία όλων των Step Definitions με όσο το δυνατόν περισσότερο κώδικα.	Ακέραια τιμή	-
4	Χρησιμοποίηση και Ακρίβεια Data Access Objects	Αξιολόγηση της ακρίβειας χρήσης και των υπενθυμίσεων για Data Access Objects από το Ai.	Χρήση DAO: 1 αν χρησιμοποιηθεί από την αρχή / 0.5 αν δεν χρησιμοποιηθεί από την αρχή Ακέραιες τιμές	Βαθμολογία χρήσης DAO = Χρήση DAO × (Ακριβείς DAO - Απαιτούμενες Υπενθυμίσεις)
5	Χρησιμοποίηση και ακρίβεια των Services	Αξιολόγηση της αποδοχής και των υπενθυμίσεων για Services από το LLM.	Χρήση Services: 1 αν χρησιμοποιηθεί από την αρχή / 0.5 αν δεν χρησιμοποιηθεί από την αρχή Ακέραιες τιμές	Βαθμολογία χρήσης Services = Χρήση Services × (Αποδεκτά Services - Απαιτούμενες Υπενθυμίσεις)
6	Ακρίβεια κλάσεων Domain που προτάθηκαν/χρησιμοποιήθηκαν	Μέτρηση του ποσοστού των κλάσεων Domain που προτάθηκαν ή χρησιμοποιήθηκαν σωστά από το LLM.	Ποσοστιαία τιμή	Ποσοστό ακρίβειας κλάσεων Domain = (Αριθμός σωστών προτεινόμενων Κλάσεων Domain / Συνολικός Αριθμός Κλάσεων Domain) × 100
7	Αποδεκτές λύσεις Step Definition	Μέτρηση του ποσοστού των Step Definitions που είναι αποδεκτές βάσει της ολοκλήρωσης, της ακρίβειας, της λειτουργικότητας και της ενσωμάτωσης.	Ποσοστιαία τιμή	Ποσοστό αποδεκτών ορισμών Step Definitions = (Αριθμός Αποδεκτών Step Definitions / Συνολικός Αριθμός Step Definitions) × 100

8	Καλύτερες από αποδεκτές λύσεις Step Definitions	Μέτρηση του ποσοστού των Step Definitions που είχαν καλύτερες από αποδεκτές λύσεις.	Ποσοστιαία τιμή	Ποσοστό καλύτερων από αποδεκτές λύσεων = (Αριθμός Καλύτερων από Αποδεκτές Step Definitions / Συνολικός Αριθμός Step Definitions) × 100
9	Αντικατάσταση αντικειμένων σε φυσική γλώσσα με μεταβλητές κώδικα	Περιγραφή εάν το LLM αντικαθιστά περιπτώσεις αντικειμένων που εκφράζονται σε φυσική γλώσσα με μεταβλητές στον κώδικα.	Αλφαριθμητικό	-
10	Ακρίβεια ιδιοτήτων	Μέτρηση του ποσοστού των σωστών ιδιοτήτων που χρησιμοποιεί το LLM αφότου του έχουν δοθεί (φάση 3,4)	Ποσοστιαία τιμή	Βαθμολογία ακρίβειας ιδιοτήτων= (Σωστές ιδιότητες / (Σωστές ιδιότητες + Αγνοούμενες ιδιότητες)) × 100
11	Ακρίβεια Μεθόδων (εξαιρουμένων των Getters και Setters)	Μέτρηση του ποσοστού των σωστών μεθόδων που χρησιμοποιεί το LLM αφότου του έχουν δοθεί (φάση 4)	Ποσοστιαία τιμή	Βαθμολογία ακρίβειας μεθόδων = (Σωστές Μέθοδοι) / (Σωστές Μέθοδοι + Αγνοούμενοι Μέθοδοι) × 100
12	Επανάληψη/Πρόσθετη επεξήγηση	Καταμέτρηση των περιπτώσεων όπου το LLM χρειαζόταν πρόσθετη εξήγηση ή επανάληψη μηνυμάτων.	Ακέραια τιμή	-
13	Κενά Step Definitions	Καταμέτρηση των περιπτώσεων όπου το LLM παρείχε κενά Step Definitions και χωρίς κώδικα	Ακέραια τιμή	-

Στην συνέχεια, ακολουθεί μία αναλυτική επεξήγηση κάθε κριτηρίου του **Πίνακα 1 :**

1. **Παροχή των features ταυτόχρονα:** Το κριτήριο αυτό εξετάζει αν τα features του συστήματος της βιβλιοθήκης δόθηκαν σε ένα μόνο μήνυμα ή αν δόθηκαν σταδιακά σε περισσότερα από ένα μηνύματα. Αν η παρουσίαση των χαρακτηριστικών έγινε σε ένα μήνυμα, η αξιολόγηση αποδίδει την τιμή «Ναι». Σε αντίθετη περίπτωση, αν τα χαρακτηριστικά δόθηκαν σε πολλά μηνύματα, αποδίδεται η τιμή «Όχι».
2. **Παροχή του κώδικα των Domain/Daos/Services από την αρχή:** Αυτό το κριτήριο εξετάζει αν στο μεγάλο γλωσσικό μοντέλο δόθηκε η εντολή να παράγει εξ αρχής τον κώδικα για τις κλάσεις Domain, τα Data Access Objects (DAOs) και τις υπηρεσίες (Services) που χρησιμοποιεί το σύστημα της βιβλιοθήκης. Αν δόθηκε η εντολή αυτή, η αξιολόγηση αποδίδει την τιμή «Ναι», διαφορετικά, εάν δεν δόθηκε κάποια επιπλέον εντολή, αποδίδει την τιμή «Όχι».

3. **Απαιτούμενα μηνύματα για ολοκληρωμένα Step Definitions:** Το κριτήριο αυτό μετρά τον αριθμό των μηνυμάτων που απαιτήθηκαν για τη δημιουργία του κώδικα για τα Step Definitions από το μεγάλο γλωσσικό μοντέλο. Εξετάζεται ο συνολικός αριθμός μηνυμάτων που ανταλλάχθηκαν στην συγκεκριμένη συνομιλία, προκειμένου να ολοκληρωθούν τα Step Definitions και ο μικρότερος αριθμός μηνυμάτων δείχνει καλύτερη αποδοτικότητα και κατανόηση από το σύστημα.
4. **Χρησιμοποίηση και ακρίβεια Data Access Objects:** Το κριτήριο αυτό μετρά την ακρίβεια του μεγάλου γλωσσικού μοντέλου να χρησιμοποιεί και να προτείνει τις σωστές κλάσεις Data Access Objects. Υπολογίζεται από τον τύπο «Βαθμολογία Χρήσης DAO = Χρήση DAO × (Ακριβείς DAO - Απαιτούμενες Υπενθυμίσεις)», όπου η μεταβλητή «χρήση DAO» έχει την τιμή 1 εάν το σύστημα χρησιμοποίησε χωρίς καμία επιπρόσθετη βοήθεια τα Data Access Objects, και 0.5 εάν χρειάστηκε κάποιο επιπλέον μήνυμα και δεν τα χρησιμοποίησε από την αρχή. Ακόμη, η μεταβλητή «Ακριβείς DAO» είναι ο αριθμός τον σωστά προτεινόμενων/χρησιμοποιημένων DAO που χρησιμοποίησε το μεγάλο γλωσσικό μοντέλο στην συνομιλία, ενώ οι «απαιτούμενες υπενθυμίσεις» είναι ένας ακέραιος αριθμός ο οποίος μετρά πόσα μηνύματα χρειάστηκαν για να υπενθυμιστούν τα DAOs στο μοντέλο. Ένα υψηλό ποσοστό ακρίβειας σημαίνει καλύτερη κατανόηση και χρήση των DAOs από το μοντέλο.
5. **Χρησιμοποίηση και ακρίβεια των Services:** Το κριτήριο αυτό μετρά την ακρίβεια του μεγάλου γλωσσικού μοντέλου να χρησιμοποιεί και να προτείνει τα σωστά Service κλάσεις, τα οποία είναι βοηθητικές κλάσεις που λειτουργούν ως υπηρεσίες για το σύστημα της βιβλιοθήκης. Υπολογίζεται από τον τύπο «Βαθμολογία Χρήσης Services = Χρήση Services × (Αποδεκτά Services - Απαιτούμενες Υπενθυμίσεις)», όπου η μεταβλητή «χρήση Services» έχει την τιμή 1 εάν το σύστημα χρησιμοποίησε χωρίς καμία επιπρόσθετη βοήθεια τα Services από την αρχή, και 0.5 εάν χρειάστηκε κάποιο επιπλέον μήνυμα για να τα χρησιμοποιήσει. Ακόμη, η μεταβλητή «Αποδεκτά Services» μετράει τον αριθμό των αποδεκτών service κλάσεων που χρησιμοποίησε το μεγάλο γλωσσικό μοντέλο, ακόμη και αν είχαν κάποιες διαφορές από τα services του πραγματικού συστήματος βιβλιοθήκης, ενώ οι «απαιτούμενες υπενθυμίσεις» είναι ένας ακέραιος αριθμός ο οποίος μετράει το πόσα μηνύματα υπενθύμισης χρειάστηκαν να δοθούν στο σύστημα για την χρήση των Services

στην υλοποίησή του. Ένα υψηλό ποσοστό υποδεικνύει σημαίνει καλή κατανόηση και χρήση των Services από το μοντέλο.

6. **Ακρίβεια κλάσεων Domain που προτάθηκαν/χρησιμοποιήθηκαν:** Το κριτήριο αυτό μετράει το ποσοστό των Domain κλάσεων του συστήματος της βιβλιοθήκης που το μεγάλο γλωσσικό μοντέλο πρότεινε και χρησιμοποίησε σωστά στην υλοποίηση του κώδικα των Step Definitions. Το ποσοστό αυτό προκύπτει από τον τύπο «Ποσοστό Ακρίβειας Κλάσεων Domain = (Αριθμός σωστών προτεινόμενων/χρησιμοποιούμενων Κλάσεων Domain / Συνολικός Αριθμός Κλάσεων Domain) × 100» όπου ο συνολικός αριθμός κλάσεων Domain είναι οι συνολικές σωστές κλάσεις που υπάρχουν στο σύστημα βιβλιοθήκης. Ένα υψηλό ποσοστό υποδεικνύει ότι το μοντέλο κατάφερε να χρησιμοποιήσει σωστά τις κλάσεις Domain.
7. **Αποδεκτές λύσεις Step Definition:** Το κριτήριο αυτό μετρά το ποσοστό των αποδεκτών λύσεων του κώδικα των Step Definitions που παρήγαγε το μεγάλο γλωσσικό μοντέλο. Αποδεκτή θεωρείται μία λύση που θα μπορούσε να ενσωματωθεί με μικρές τροποποιήσεις στο πραγματικό σύστημα βιβλιοθήκης και γενικά εάν έχει μία ορθή λογική η ο κώδικας των βημάτων. Το ποσοστό αυτό προκύπτει από τον τύπο «Ποσοστό Αποδεκτών Ορισμών Step Definitions = (Αριθμός Αποδεκτών Step Definitions / Συνολικός Αριθμός Step Definitions) × 100» όπου ο συνολικός αριθμός Step Definitions είναι τα συνολικά Step Definitions που δημιουργήθηκαν για τα σενάρια Cucumber. Ένα υψηλό ποσοστό δείχνει ότι το μοντέλο παράγει λύσεις που είναι αποδεκτές και εφαρμόσιμες.
8. **Καλύτερες από αποδεκτές λύσεις Step Definitions:** Το κριτήριο αυτό μετρά το ποσοστό των αποδεκτών λύσεων του κώδικα των Step Definitions που είναι καλύτερες από τις αποδεκτές λύσεις ή ακόμα καλύτερες από τον υπάρχοντα κώδικα. Το ποσοστό αυτό προκύπτει από τον τύπο «Ποσοστό Καλύτερων από Αποδεκτές Λύσεων = (Αριθμός Καλύτερων από Αποδεκτές Step Definitions / Συνολικός Αριθμός Step Definitions) × 100» όπου ο συνολικός αριθμός Step Definitions είναι τα συνολικά Step Definitions που δημιουργήθηκαν για τα σενάρια Cucumber. Το υψηλό ποσοστό δείχνει ότι το μοντέλο δεν παρέχει μόνο αποδεκτές λύσεις, αλλά και βελτιωμένες και αποτελεσματικές λύσεις.
9. **Αντικατάσταση αντικειμένων σε φυσική γλώσσα με μεταβλητές κώδικα:** Το κριτήριο αυτό αξιολογεί την ικανότητα του μεγάλου γλωσσικού μοντέλου να αντιληφθεί και να χρησιμοποιήσει μεταβλητές που περιγράφονται σε φυσική γλώσσα στα σενάρια Cucumber

και να τις ενσωματώσει ως μεταβλητές στον κώδικα των Step Definitions. Το μοντέλο αξιολογείται για το πόσο καλά μετατρέπει τις περιγραφές σε φυσική γλώσσα σε κατάλληλες μεταβλητές στον κώδικα. Η σωστή αντικατάσταση υποδεικνύει καλή κατανόηση της γλώσσας και του κώδικα που απαιτείται για την παραγωγή των Step Definitions.

10. **Ακρίβεια ιδιοτήτων:** Το κριτήριο αυτό αξιολογεί την ικανότητα του μεγάλου γλωσσικού μοντέλου να χρησιμοποιεί/προτείνει τις σωστές ιδιότητες όλων των Domain κλάσεων, αφότου του έχουν δοθεί ως πληροφορίες. Για τον λόγο αυτό, το κριτήριο αυτό χρησιμοποιείται μόνο την φάση 3 και στην φάση 4 όπου παρέχουμε τις πληροφορίες αυτές στο μεγάλο γλωσσικό μοντέλο. Η ικανότητα αυτή μετριέται σε ποσοστό και δίνεται από τον τύπο «Βαθμολογία Ακρίβειας ιδιοτήτων= (Σωστές ιδιότητες / (Σωστές ιδιότητες + Αγνοούμενες ιδιότητες)) × 100» όπου η μεταβλητή «Σωστές ιδιότητες» είναι ένας ακέραιος αριθμός με τον συνολικό αριθμό των σωστών ιδιοτήτων που μάντεψε το σύστημα, ενώ «Αγνοούμενες ιδιότητες» είναι οι ιδιότητες που δόθηκαν αλλά παραλήφθηκαν να χρησιμοποιηθούν από το σύστημα. Ένα υψηλό ποσοστό υποδεικνύει ότι το μοντέλο κατανόησε τις πληροφορίες που δέχθηκε και μπόρεσε να τις χρησιμοποιήσει στην υλοποίησή του.
11. **Ακρίβεια μεθόδων (εξαιρούμενων των Getters και Setters):** Το κριτήριο αυτό εξετάζει την ικανότητα του μεγάλου γλωσσικού μοντέλου να χρησιμοποιεί/προτείνει σωστά τις μεθόδους όλων των Domain κλάσεων, αφότου του έχουν δοθεί ως πληροφορίες. Για τον λόγο αυτό, το κριτήριο αυτό χρησιμοποιείται μόνο στην φάση 4, στην οποία παρέχουμε ως πληροφορία τις μεθόδους που αποτελούν κάθε Domain κλάση. Η ικανότητα αυτή μετριέται σε ποσοστό και δίνεται από τον τύπο «Βαθμολογία Ακρίβειας Μεθόδων = (Σωστές Μέθοδοι) / (Σωστές Μέθοδοι + Αγνοούμενοι Μέθοδοι) × 100» όπου η μεταβλητή «Σωστές Μέθοδοι» είναι ένας ακέραιος αριθμός με τον συνολικό αριθμό σωστών μεθόδων που μάντεψε το σύστημα, ενώ «Αγνοούμενοι μέθοδοι» είναι οι μέθοδοι που δόθηκαν αλλά παραλήφθηκαν να χρησιμοποιηθούν από το σύστημα. Ένα υψηλό ποσοστό υποδεικνύει ότι το μοντέλο κατανόησε τις μεθόδους που δέχθηκε και μπόρεσε να τις χρησιμοποιήσει στην υλοποίησή του.
12. **Επανάληψη/Πρόσθετη επεξήγηση:** Το κριτήριο αυτό μετρά τον αριθμό των μηνυμάτων που απαιτήθηκε να δοθούν στο μεγάλο γλωσσικό μοντέλο ως υπενθύμιση, επανάληψη ή

βοήθεια για να συνεχίσει να παράγει. Ένα χαμηλό νούμερο στο κριτήριο αυτό υποδεικνύει ότι σύστημα είχε καλύτερη κατανόηση των πληροφοριών που το δόθηκαν.

13. **Κενά Step Definitions:** Το κριτήριο αυτό μετρά τον αριθμό των φορών που το μεγάλο γλωσσικό μοντέλο παρήγαγε Step Definitions χωρίς κάποιο κώδικα ή με πολύ λίγη πληροφορία για το Step Definition. Ένα χαμηλό νούμερο στο κριτήριο αυτό υποδεικνύει ότι το σύστημα παρήγαγε ευκολότερα κώδικα και με πιο αποτελεσματικό τρόπο.

Τα κριτήρια αξιολόγησης που παρουσιάζονται παραπάνω παρέχουν μια ολοκληρωμένη βάση για την ανάλυση της απόδοσης των μεγάλων γλωσσικών μοντέλων που χρησιμοποιήθηκαν στην εργασία. Καλύπτουν κρίσιμες πτυχές της χρησιμότητας των μοντέλων, προσφέροντας ένα μεθοδικό εργαλείο για τη σύγκριση και την αξιολόγηση των δυνατοτήτων τους στην παραγωγή αυτοματοποιημένου κώδικα για τα σενάρια BDD.

5.3 Εμπειρική αξιολόγηση

Όπως αναφέρθηκε αναλυτικά και στην ενότητα της μεθοδολογίας, η αξιολόγηση των μεγάλων γλωσσικών μοντέλων έγινε σε τέσσερις φάσεις, όπου κάθε φάση αντιπροσωπεύει διαφορετική ποσότητα γνώσης που παρέχεται στα μοντέλα.

5.3.1 Αξιολόγηση της φάσης 1

Ο **Πίνακας 2** που φαίνεται παρακάτω παρουσιάζει τα αποτελέσματα της **πρώτης φάσης** της αξιολόγησης της απόδοσης των μεγάλων γλωσσικών μοντέλων στην παραγωγή αυτοματοποιημένου κώδικα για τα σενάρια BDD. Στις σειρές του πίνακα παρουσιάζονται τα διαφορετικά αριθμημένα κριτήρια που αναλύθηκαν παραπάνω και στις στήλες του πίνακα οι διαφορετικές συζητήσεις που έγιναν με τα διαφορετικά μεγάλα γλωσσικά μοντέλα.

Πίνακας 2. Φάση 1 αξιολόγησης των μεγάλων γλωσσικών μοντέλων

Φάση 1															
	GPT -3.5				GitHub COPILOT 3.5 Turbo				GPT -4				GPT -4o		
Αριθμός Κριτηρίου	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 4	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 4	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 4	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3
1	Ναι	Ναι	Όχι	Όχι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Όχι	Όχι	Ναι	Ναι	Όχι
2	Όχι	Ναι	Ναι	Ναι	Όχι	Όχι	Όχι	Ναι	Όχι	Ναι	Όχι	Ναι	Όχι	Ναι	Όχι
3	13	10	11	11	8	7	9	8	9	8	8	8	5	7	5
4	3	1	2	1	0	1	3	3	0.5	3	1	2	2	2	2
5	3	3	2	3	3	3	3	4	3	3	3	4	3	3	4
6	28.57 %	42.85%	28.57%	42.85%	28.57%	42.85 %	42.85 %	42.85 %	28.57%	42.85 %	28.57%	28.57%	28.57%	28.57%	28.57%
7	22.91%	31.25%	20.83%	20.83%	20.83%	33.3%	18.75%	22.91%	18.75%	33.3%	8%	10%	35.4%	20,8%	18%
8	0%	0%	0%	0%	0%	0%	0%	0%	0%	2%	0%	0%	2%	0%	0%
9	Το σύστημα δεν χρησιμοποίησε πλήρως τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα, μόνο σε λίγα Step Definitions	Το σύστημα δεν χρησιμοποίησε πλήρως τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα, μόνο σε λίγα Step Definitions	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το LLM κατάλαβε και χρησιμοποίησε εν μέρει κάποια αντικείμενα που εκφράστηκαν σε φυσική γλώσσα, αλλά όχι σε όλα τα Step Definitions	Το LLM κατανόησε σχεδόν σε όλα τα Step Definitions τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα.	Το LLM κατανόησε σχεδόν σε όλα τα Step Definitions τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα.	Το LLM κατανόησε σχεδόν σε όλα τα Step Definitions τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα.	Το LLM χρησιμοποίησε μόνο λίγα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα.	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το LLM κατάλαβε και χρησιμοποίησε εν μέρει κάποια αντικείμενα που εκφράστηκαν σε φυσική γλώσσα, αλλά όχι σε όλα τα Step Definitions	Το LLM κατάλαβε και χρησιμοποίησε εν μέρει κάποια αντικείμενα που εκφράστηκαν σε φυσική γλώσσα, αλλά όχι σε όλα τα Step Definitions	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα
12	4	4	4	4	4	2	3	3	5	2	2	2	0	1	0
13	28	7	15	11	23	11	19	8	10	4	2	2	0	10	0

Στην αρχική αυτή φάση της αξιολόγησης, παρατηρήθηκε ότι όλα τα μεγάλα γλωσσικά μοντέλα παρουσίασαν περιορισμένα αποτελέσματα σε σχέση με την ποσότητα του παραγόμενου κώδικα, τη λεπτομέρεια των απαντήσεων και την αποδοχή των παραγόμενων Step Definitions. Συγκεκριμένα, στις αρχικές συζητήσεις όπου δεν δόθηκε εντολή δημιουργίας του κώδικα για τα Domain/Data Access Objects/Services, τα μοντέλα αντιμετώπισαν σημαντικές δυσκολίες στην παραγωγή κώδικα. Αυτό είχε ως αποτέλεσμα την εμφάνιση πολλών κενών Step Definitions και

την ανάγκη για πολλές υπενθυμίσεις ή εντολές προς το σύστημα για να παραχθεί ο απαιτούμενος κώδικας. Στον πίνακα εμφανίζεται ότι ο αριθμός των κενών Step Definitions έφτασε έως και 28.

Η κατάσταση αυτή βελτιώθηκε σημαντικά όταν αρχίσαμε να δίνουμε την εντολή στα μοντέλα να παράγουν πρώτα τον κώδικα των Domain/Data Access Objects/Services, προτού προχωρήσουν στην παραγωγή των Step Definitions. Αυτή η προσέγγιση βοήθησε τα μοντέλα να εστιάσουν καλύτερα και να χρησιμοποιήσουν αποτελεσματικότερα τις κλάσεις που είχαν ήδη δημιουργήσει.

Ως αποτέλεσμα, παρατηρήθηκε σημαντική μείωση στον αριθμό των κενών Step Definitions και βελτίωση στην ποιότητα των παραγόμενων Step Definitions, με πολλά μοντέλα να επιδεικνύουν αυξημένα ποσοστά αποδεκτών Step Definitions ή να διατηρούν υψηλά ποσοστά αποδοχής. Επιπλέον, η τεχνική αυτή φάνηκε να ενίσχυσε τη χρήση των Data Access Objects από τα μοντέλα, κάτι που είχε παραληφθεί σε πολλές προηγούμενες περιπτώσεις. Ειδικότερα, τα GitHub Copilot και GPT-4o ξεχώρισαν καθώς παρήγαγαν από μόνα τους τον κώδικα των Domain/Data Access Objects/Services χωρίς επιπλέον εντολές, υποδεικνύοντας την ευεργετική επίδραση αυτής της τεχνικής στην απόδοση των μοντέλων. Τα δύο αυτά μοντέλα επίσης αποδείχθηκαν ιδιαίτερα ικανά στην κατανόηση και τη χρήση μεταβλητών σε φυσική γλώσσα, κάτι που συνέβαλε στην παραγωγή πιο ακριβούς κώδικα.

Στη συνέχεια, δοκιμάστηκε η μέθοδος της σταδιακής παρουσίασης των απαιτήσεων, όπου κάθε feature δινόταν σε ξεχωριστό μήνυμα, με σκοπό να επικεντρωθεί το μοντέλο σε κάθε feature ξεχωριστά και να βελτιώσει τα αποτελέσματα. Ωστόσο, η παρατήρηση έδειξε ότι η σταδιακή παρουσίαση δεν βελτίωσε την απόδοση σε σύγκριση με άλλες τεχνικές, καθώς τα αποδεκτά Step Definitions συχνά ήταν λιγότερα και τα μοντέλα έχαναν την ευρύτερη εικόνα του συστήματος που θα μπορούσε να οδηγήσει σε πιο ακριβή αποτελέσματα.

Από την πρώτη φάση, προκύπτει ότι το GitHub Copilot υπερείχε των GPT-3.5 και GPT-4 σε πολλές περιπτώσεις, καθώς παρουσίασε καλύτερη κατανόηση των δεδομένων και ταχύτερη παραγωγή κώδικα με λιγότερα κενά Step Definitions. Το GPT-4o, ως το πιο πρόσφατο και προηγμένο μοντέλο, αποδείχθηκε ανώτερο από όλα τα άλλα μοντέλα, επιδεικνύοντας καλύτερη κατανόηση των πληροφοριών, ελάχιστο αριθμό μηνυμάτων για την παραγωγή των Step Definitions και ταχύτητα παραγωγής κώδικα που ξεπέρασε όλα τα άλλα μοντέλα.

5.3.2 Αξιολόγηση της φάσης 2

Στη **δεύτερη φάση** του πειράματος, χρησιμοποιήθηκαν οι ίδιες τεχνικές παρουσίασης της πληροφορίας όπως και στην πρώτη φάση, και παρουσιάζονται αναλυτικά στον **Πίνακα 3** παρακάτω. Ωστόσο, τα αποτελέσματα ήταν κατά μεγάλο ποσοστό παρόμοια ή και χειρότερα σε πολλές περιπτώσεις. Ειδικότερα, παρατηρήθηκε ότι τα ποσοστά των αποδεκτών Step Definitions ήταν πολύ χαμηλότερα σε σύγκριση με την πρώτη φάση, παρά το γεγονός ότι τα γλωσσικά μοντέλα είχαν πλέον περισσότερη γνώση. Αυτό υποδηλώνει ότι τα ονόματα των κλάσεων δεν συμβάλλουν ουσιαστικά στη βελτίωση των αποτελεσμάτων.

Στις περισσότερες περιπτώσεις, τα μοντέλα αδυνατούσαν να κατανοήσουν τις συνδέσεις μεταξύ των κλάσεων και περιορίζονταν στη χρήση μόνο των πιο βασικών κλάσεων, παραλείποντας πολλές άλλες. Όπως και στην προηγούμενη φάση, η εντολή για την παραγωγή των Domain/Data Access Objects/Services συνέβαλε σημαντικά στη βελτίωση των απαντήσεων των γλωσσικών μοντέλων. Ειδικότερα, αυτή η τεχνική βοήθησε τα μοντέλα να κατανοήσουν καλύτερα και να παράγουν πιο αποδεκτά Step Definitions, ενώ διευκόλυνε επίσης τη χρήση των Data Access Objects για την αποθήκευση και εύρεση αντικειμένων, όπως φαίνεται και από τα ποσοστά χρήσης ορθών Data Access Objects.

Η τεχνική της παρουσίασης των απαιτήσεων σε φυσική γλώσσα σε ξεχωριστά μηνύματα δημιούργησε παρόμοια προβλήματα με εκείνα της πρώτης φάσης. Αν και υπήρξαν κάποιες περιπτώσεις με θετικά αποτελέσματα, η συνολική απόδοση δεν δικαίωσε την τεχνική αυτή ως την καλύτερη επιλογή. Αντιθέτως, η παρουσίαση των απαιτήσεων σε ένα μόνο μήνυμα και η εντολή για παραγωγή του κώδικα των Domain/Data Access Objects/Services πρώτα παρέμειναν πιο αποτελεσματικές.

Συμπερασματικά, το GitHub Copilot συνεχίζει να ξεχωρίζει σε σχέση με τα GPT-3.5 και GPT-4, αποδεικνύοντας ότι κατανοεί καλύτερα τις μεταβλητές που έχουν δοθεί σε φυσική γλώσσα, όπως 'George Red', 'Moby Dick' και 'Harry Potter', και τις χρησιμοποιεί σωστά σε πολλές περιπτώσεις. Ωστόσο, το GPT-4o επιβεβαιώνεται για άλλη μία φορά ως το ανώτερο μοντέλο, παράγοντας πολύ καλύτερα αποτελέσματα από τα υπόλοιπα γλωσσικά μοντέλα. Το GPT-4o χρησιμοποίησε αποδοτικά την επιπλέον γνώση από τα ονόματα των Domain κλάσεων και παρήγαγε πολλά και υψηλής ποιότητας Step Definitions, όπως φαίνεται από τα αποτελέσματα στο κριτήριο αξιολόγησης 8.

Πίνακας 3. Φάση 2 αξιολόγησης των μεγάλων γλωσσικών μοντέλων

Φάση 2															
	GPT -3.5			**GitHub COPILOT 3.5 Turbo**					**GPT -4**				**GPT -4o**		
Αριθμός Κριτηρίου	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 4	Συνομιλία 5	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 4	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3
1	Ναι	Ναι	Όχι	Ναι	Ναι	Ναι	Ναι	Όχι	Ναι	Ναι	Όχι	Όχι	Ναι	Ναι	Ναι
2	Ναι	Όχι	Όχι	Όχι	Όχι	Ναι	Ναι	Ναι	Όχι	Ναι	Όχι	Ναι	Όχι	Ναι	Ναι
3	10	10	7	9	11	10	8	7	9	9	9	8	6	3	5
4	3	0	0	0	3	0	3	1	3	2	1	1	3	3	3
5	3	2	3	3	3	3	3	2	3	4	3	3	3	1	3
7	6.25%	2.08%	10.4%	20.8%	12.5%	15%	33.3%	20.8%	25%	33.3%	12.5%	31.2%	50%	54.1%	37.5%
8	0%	0%	0%	0%	0%	0%	2%	2%	2%	2%	2%	8.33%	20.83%	20.83%	8.33%
9	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το LLM δεν κατανόησε πολύ καλά τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε μόνο σε σπάνιες περιπτώσεις .	Το LLM δεν κατανόησε πολύ καλά τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε μόνο σε σπάνιες περιπτώσεις.	Το LLM δεν κατανόησε πολύ καλά τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε μόνο σε σπάνιες περιπτώσεις.	Το LLM δεν κατανόησε πολύ καλά τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε μόνο σε σπάνιες περιπτώσεις.	Το LLM κατανόησε μόνο εν μέρει τα αντικείμενα που δόθηκαν σε φυσική γλώσσα, χάνοντας πολλά από αυτά.	Το LLM κατανόησε άψογα τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το LLM δεν κατανόησε ούτε χρησιμοποίησε τα αντικείμενα που δόθηκαν σε φυσική γλώσσα.
12	4	3	1	1	5	2	1	1	2	2	3	2	0	0	0
13	31	26	26	3	15	0	9	5	1	1	1	3	6	4	0

5.3.3 Αξιολόγηση της φάσης 3

Στην **τρίτη φάση** του πειράματος, επεκτείνεται η γνώση που παρέχουμε στα γλωσσικά μοντέλα, και όπως αναφέρθηκε και στην υποενότητα της μεθοδολογίας, προσθέτουμε νέες πληροφορίες πέρα από την αρχιτεκτονική του συστήματος, τις γενικές πληροφορίες σχετικά με το σύστημα, τις απαιτήσεις σε φυσική γλώσσα και τα ονόματα των Domain κλάσεων. Αυτή τη φορά, συμπληρώνουμε την πληροφορία με τις ιδιότητες κάθε κλάσης, δίνοντας στα μοντέλα τη

δυνατότητα να ανακαλύψουν και να κατανοήσουν καλύτερα τις σχέσεις μεταξύ των κλάσεων και τις μεθόδους που πρέπει να εφαρμόσουν.

Η προσθήκη αυτών των ιδιοτήτων, όπως περιγράφεται και από τον **Πίνακα 4** παρακάτω, φαίνεται να έχει σημαντική επίδραση στην απόδοση των γλωσσικών μοντέλων. Τα μοντέλα βελτίωσαν σημαντικά την ανάλυση και τη χρήση των κλάσεων, όπως δείχνουν τα αυξημένα ποσοστά στα αποδεκτά Step Definitions και τα Step Definitions που ξεπέρασαν την αποδεκτή λύση. Η εντολή για την παραγωγή των Domain/Data Access Objects/Services συνεχίζει να αποδεικνύεται χρήσιμη, με τα γλωσσικά μοντέλα να κατανοούν και να παράγουν κώδικα πιο αποτελεσματικά. Αυτό αποτυπώνεται στην αύξηση των ποσοστών στις αποδοτικές συναρτήσεις και τη μείωση των κενών Step Definitions.

Σε αυτή τη φάση, παρατηρείται ότι η τεχνική της παρουσίασης των απαιτήσεων σε φυσική γλώσσα σε μεμονωμένα μηνύματα χρησιμοποιείται όλο και λιγότερο. Τα αποτελέσματα από τις προηγούμενες φάσεις δεν επιβεβαιώνουν ότι αυτή η τεχνική προσφέρει τα καλύτερα αποτελέσματα, με αποτέλεσμα οι συνομιλίες να επικεντρώνονται περισσότερο στις άλλες αναφερθείσες τεχνικές.

Το GitHub Copilot παρουσιάζει σημαντική βελτίωση σε σύγκριση με τα GPT-3.5 και GPT-4, ειδικότερα όσον αφορά τα αποδεκτά Step Definitions, τα καλύτερα από τα αποδεκτά Step Definitions, και την κατανόηση των μεταβλητών που δίνονται σε φυσική γλώσσα. Από την άλλη, το GPT-4o συνεχίζει να ξεχωρίζει, παρέχοντας πλήρη κώδικα με ελάχιστα μηνύματα, χωρίς κενά, και με εξαιρετική κατανόηση των δεδομένων και των ιδιοτήτων που του έχουν παραχωρηθεί.

Επιπλέον, τα τέσσερα γλωσσικά μοντέλα δείχνουν σημαντική πρόοδο στην κατανόηση των ορθών Data Access Objects και Services, σε σύγκριση με τις προηγούμενες φάσεις, όπως αποδεικνύεται από τα κριτήρια αξιολόγησης 4 και 5.

Πίνακας 4. Φάση 3 αξιολόγησης των μεγάλων γλωσσικών μοντέλων

Φάση 3														
	GPT -3.5				GitHub COPILOT 3.5 Turbo				GPT -4			GPT -4o		
Αριθμός Κριτηρίου	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 4	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 4	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3
1	Ναι	Ναι	Ναι	Όχι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι
2	Όχι	Όχι	Ναι	Ναι	Όχι	Όχι(το έκανε από μόνο του)	Ναι	Ναι	Όχι	Ναι	Όχι	Όχι	Ναι	Ναι
3	8	11	9	11	6	11	9	9	11	7	11	4	4	3
4	1	0	3	2	2	2	0	3	2	3	1	3	3	3
5	2	0.5	4	3	3	3	1	3	3	4	4	3	3	2
7	6.25%	18.75%	37.5%	35.41%%	54.1%	56.25%	22.91%	20.8%	22.91%	37.5%	20.8%	68.75%	60.41%	56.25%
8	0%	0%	4.1%	4.1%	4.1%	18.75%	14%	2%	4.1%	10.41%	4.1%	22.91%	20.83%	14.58%
9	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το LLM κατανόησε εν μέρει τα αντικείμενα που δόθηκαν σε φυσική γλώσσα. 4ο	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το LLM κατανόησε σε κάποιο βαθμό τα αντικείμενα που δόθηκαν σε φυσική γλώσσα, αλλά δεν τα χρησιμοποίησε σε όλα τα Step Definitions	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το LLM κατανόησε σε κάποιο βαθμό τα αντικείμενα που δόθηκαν σε φυσική γλώσσα, αλλά δεν τα χρησιμοποίησε σε όλους τους ορισμούς βημάτων.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το LLM κατανόησε σε κάποιο βαθμό τα αντικείμενα που δόθηκαν σε φυσική γλώσσα, αλλά δεν τα χρησιμοποίησε σε όλους τους ορισμούς βημάτων..	Το LLM κατανόησε σε κάποιο βαθμό τα αντικείμενα που δόθηκαν σε φυσική γλώσσα, αλλά δεν τα χρησιμοποίησε σε όλους τους ορισμούς βημάτων.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε σωστά.
10	25%	15.6%	31.25%	43.75%	42.85%	59.37%	62,5%	31.25%				71.87%	78.1%	68.75%
12	1	3	2	4	0	4	1	2	2	2	2	0	0	0
13	24	4	7	38	6	12	0	3	1	1	2	0	2	9

5.3.4 Αξιολόγηση της φάσης 4

Στην τελευταία φάση του πειράματος, τη **φάση 4**, όπως αναφέρθηκε προηγουμένως, επεκτείνουμε τη γνώση που παρέχουμε στα γλωσσικά μοντέλα με την εισαγωγή επιπλέον πληροφοριών. Εκτός από την αρχιτεκτονική του συστήματος, τις γενικές πληροφορίες, τις απαιτήσεις σε φυσική γλώσσα, τα ονόματα των Domain κλάσεων και τις ιδιότητες κάθε κλάσης, προσθέτουμε και τα

ονόματα όλων των μεθόδων κάθε κλάσης, συμπεριλαμβανομένων των τύπων επιστροφής και των παραμέτρων τους. Αυτό επιτρέπει στα μοντέλα να έχουν την καλύτερη δυνατή γνώση για να παράγουν αποδοτικά και αποδεκτά αποτελέσματα στα αυτοματοποιημένα τεστ.

Η προσθήκη αυτών των πληροφοριών, όπως φαίνεται και στον **Πίνακα 5** παρακάτω, έχει θετική επίδραση στην απόδοση των γλωσσικών μοντέλων. Παρατηρήθηκε ότι, σε σχεδόν κάθε συζήτηση με κάθε γλωσσικό μοντέλο, οι κλάσεις Data Access Objects και Services κατανοήθηκαν πλήρως, κάτι που είχε αρχίσει να παρατηρείται στην προηγούμενη φάση αλλά σε μικρότερο βαθμό. Τα ποσοστά των αποδεκτών Step Definitions και των καλύτερων από τα αποδεκτά Step Definitions αυξήθηκαν σημαντικά, ανεξαρτήτως του τρόπου παρουσίασης των πληροφοριών, δείχνοντας ότι η προσθήκη αυτών των στοιχείων ήταν αναγκαία για τη βελτίωση της απόδοσης του συστήματος.

Η εντολή για την παραγωγή του κώδικα των Domain/Data Access Objects/Services πρώτα συνέχισε να βελτιώνει την απόδοση σε ορισμένες περιπτώσεις, αν και δεν παρατηρήθηκε σημαντική διαφορά σε σχέση με τις συνομιλίες όπου η εντολή αυτή παραλείφθηκε, καθώς το σύστημα συχνά παρήγαγε αυτόν τον κώδικα αυτόματα. Η τάση του συστήματος να παράγει πρώτα τον κώδικα των Domain/Data Access Objects/Services επιβεβαιώθηκε ξανά, συμβάλλοντας στην αποτελεσματική παραγωγή του αυτοματοποιημένου κώδικα.

Η τεχνική της παρουσίασης των απαιτήσεων σε ξεχωριστά μηνύματα παραλείφθηκε και σε αυτή τη φάση, καθώς αποδείχθηκε ότι έχει περισσότερα μειονεκτήματα από τα οφέλη της.

Το GitHub Copilot ξεχώρισε για την ικανότητά του να κατανοεί καλύτερα από τα GPT-3.5 και GPT-4 κάθε κριτήριο, αν και, αν και είχε μικρό ποσοστό καλύτερων από τα αποδεκτά Step Definitions, δεν έφτασε στα σχεδόν τέλεια αποτελέσματα του GPT-4o. Το GPT-4o, με τις νέες πληροφορίες, παρουσίασε εξαιρετικά αποτελέσματα με ελάχιστα μηνύματα, καταδεικνύοντας μεγάλη ταχύτητα και αποδοτικότητα, και σχεδόν τέλεια ποσοστά στην χρήση των ιδιοτήτων και συναρτήσεων που του δόθηκαν (κριτήρια 10 και 11).

Συνολικά, τα γλωσσικά μοντέλα GitHub Copilot και GPT-4o είναι τα μόνα που κατανοούν σε μεγάλο βαθμό τη χρήση μεταβλητών σε φυσική γλώσσα και τις εφαρμόζουν σωστά στα αυτοματοποιημένα τεστ.

Πίνακας 5. Φάση 4 αξιολόγησης των μεγάλων γλωσσικών μοντέλων

Φάση 4													
	GPT -3.5				GitHub COPILOT 3.5 Turbo			GPT -4			GPT -4o		
Αριθμός Κριτηρίου	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 4	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3	Συνομιλία 1	Συνομιλία 2	Συνομιλία 3
1	Ναι	Ναι	Ναι	Όχι	Ναι	Ναι	Ναι	Ναι	Ναι	Όχι	Ναι	Ναι	Ναι
2	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Όχι	Ναι	Όχι	Όχι	Ναι	Όχι(το έκανε μόνο του)	Όχι(το έκανε μόνο του)
3	14	10	14	14	11	8	8	10	12	8	4	4	3
4	3	3	3	1	3	3	1	3	2	2	3	3	3
5	3	2	3	2	2	3	3	3	3	3	3	4	2
7	10.41%	37.5%	37.5%	33.3%	33.3%	52.08%	43.75%	43.75%	27.07%	31.25%	58.33%	72.91%	64.58%
8	0%	16.6%	12.5%	10.41%	14.5%	12.51%	14.5%	14.5%	4.1%	6.2%	20.83%	31.25%	27.08%
9	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το LLM κατανόησε σε κάποιο βαθμό τα αντικείμενα που παρέχονται σε φυσική γλώσσα, αλλά δεν τα χρησιμοποίησε σωστά σε όλα τα Step Definitions	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το LLM χρησιμοποίησε τα αντικείμενα που δόθηκαν σε φυσική γλώσσα μόνο μερικές φορές, αλλά όχι τόσο καλά ώστε να αξίζει να σημειωθεί.	Το LLM κατανόησε τα αντικείμενα που δόθηκαν σε φυσική γλώσσα σε πολλές περιπτώσεις, αλλά όχι σε όλες.	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το LLM κατανόησε σε κάποιο βαθμό τα αντικείμενα που παρέχονται σε φυσική γλώσσα, αλλά δεν τα χρησιμοποίησε σωστά σε όλα τα Step Definitions.	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	Το σύστημα δεν χρησιμοποίησε καθόλου τα αντικείμενα που εκφράστηκαν σε φυσική γλώσσα	TTo LLM κατανόησε σε κάποιο βαθμό τα αντικείμενα που παρέχονται σε φυσική γλώσσα, αλλά δεν τα χρησιμοποίησε σωστά σε όλους τους ορισμούς βημάτων.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε πολύ καλά.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε πολύ καλά.	Το LLM κατανόησε τέλεια τα αντικείμενα που δόθηκαν σε φυσική γλώσσα και τα χρησιμοποίησε πολύ καλά.
10	18.75%	78.12%	52.12%	46.87%	25%	46.8%	46.8%	46.8%	46.8%	43.75%	90.6%	90.6%	90.6%
11	11%	46.1%	34.6%	34.6%	30.7%	42.3%	30.7%	26.9%	34.6%	34.6%	50%	34.6%	50%
12	5	4	5	6	4	1	2	3	2	1	0	0	0
13	14	1	10	9	20	0	1	2	0	1	6	4	3

6. Συμπεράσματα

Ολοκληρώνοντας, η ανάλυση της χρήσης των μεγάλων γλωσσικών μοντέλων στα πλαίσια του BDD αποδεικνύει ότι τα εργαλεία αυτά μπορούν να προσφέρουν πληθώρα πλεονεκτήματα στον τομέα της ανάπτυξης λογισμικού και ειδικότερα στην δημιουργία αυτοματοποιημένου ελέγχου κώδικα. Συγκεκριμένα, όπως ερευνήθηκε και στην εργασία, τα μεγάλα γλωσσικά μοντέλα έχουν αποδειχθεί εξαιρετικά χρήσιμα στην αυτοματοποίηση της δημιουργίας σεναρίων BDD, στη βελτίωση της ποιότητας του κώδικα, και στη μείωση των σφαλμάτων. Ωστόσο, η έρευνα έχει επίσης αναδείξει περιορισμούς στην δημιουργία του αυτοματοποιημένου κώδικα, όπως η ανάγκη για περαιτέρω κατανόηση των δεδομένων.

Τα αποτελέσματα της μελέτης και του πειράματος που διεξήχθη, δείχνουν καθαρά ότι η εφαρμογή των μεγάλων γλωσσικών μοντέλων μπορεί να βελτιώσει σημαντικά την διαδικασία της δημιουργίας αυτοματοποιημένου κώδικα μέσω BDD, αλλά και να παρουσιάσει ορισμένα περιορισμένα αποτελέσματα που απαιτούν βελτίωση.

Στην πρώτη φάση, παρατηρήθηκε ότι τα γλωσσικά μοντέλα, παρά τη δυνατότητά τους να παράγουν κώδικα, παρουσίασαν περιορισμένα αποτελέσματα όσον αφορά την ποσότητα, την λεπτομέρεια και την αποδοχή των παραγόμενων Step Definitions. Ειδικότερα, παρατηρήθηκαν πολλά κενά Step Definitions, **ιδιαίτερα** όταν δεν δόθηκε αρχική εντολή για την παραγωγή του κώδικα των Domain/Data Access Objects/Services. Η εφαρμογή της εντολής αυτής βοήθησε στην καλύτερη εστίαση των μοντέλων και στη βελτίωση των αποτελεσμάτων, ενώ το GitHub Copilot και το GPT-4o ξεχώρισαν για την ικανότητά τους στην κατανόηση και χρήση μεταβλητών σε φυσική γλώσσα.

Στη δεύτερη φάση, αν και χρησιμοποιήθηκαν οι ίδιες τεχνικές παρουσίασης της πληροφορίας, τα αποτελέσματα δεν παρουσίασαν σημαντική βελτίωση σε σχέση με την πρώτη φάση. Τα ποσοστά αποδεκτών Step Definitions ήταν χαμηλότερα, και οι τεχνικές όπως η σταδιακή παρουσίαση δεν αποδείχθηκαν πιο αποτελεσματικές. Ωστόσο, η εντολή για παραγωγή του κώδικα των Domain/Data Access Objects/Services συνέχισε να **βελτιώνει** τα αποτελέσματα.

Στην τρίτη φάση, η προσθήκη ιδιοτήτων κλάσεων αποδείχθηκε ότι είχε θετική επίδραση στην απόδοση των μοντέλων. Τα μεγάλα γλωσσικά μοντέλα παρουσίασαν βελτιωμένη κατανόηση και χρήση των κλάσεων, οδηγώντας σε υψηλότερα ποσοστά αποδεκτών Step Definitions. Εντούτοις,

η τεχνική της παρουσίασης απαιτήσεων σε ξεχωριστά μηνύματα παρέμεινε λιγότερο αποτελεσματική.

Στην τελευταία φάση, η προσθήκη ονομάτων μεθόδων, τύπων επιστροφής και παραμέτρων κλάσεων είχε θετική επίδραση στην απόδοση των μοντέλων. Τα GitHub Copilot και GPT-4o ξεχώρισαν και πάλι για την ικανότητά τους να κατανοούν και να χρησιμοποιούν τις μεταβλητές σε φυσική γλώσσα, ενώ το GPT-4o παρουσίασε σχεδόν τέλεια αποτελέσματα με υψηλή ταχύτητα και αποδοτικότητα.

Συνολικά, τα γλωσσικά μοντέλα GitHub Copilot και GPT-4o ξεχώρισαν ως τα πιο αποτελεσματικά στην παραγωγή κώδικα για σενάρια BDD, με το GPT-4o να παρουσιάζει τα καλύτερα αποτελέσματα. Οι προσθήκες επιπλέον πληροφοριών, όπως οι ιδιότητες κλάσεων και τα ονόματα μεθόδων, βελτίωσαν την απόδοση, ενώ η τεχνική της δημιουργίας πρώτα του κώδικα των domain /Data Access Objects / Services φαίνεται να είναι η αποτελεσματικότερη τεχνική επικοινωνίας με τα μεγάλα γλωσσικά μοντέλα για παραγωγή κώδικα.

7. Βιβλιογραφία

Cucumber. (χ.χ.). *Gherkin Reference*. Ανάκτηση από Cucumber documentation: https://cucumber.io/docs/gherkin/reference/

Γιακουμάκης Ε. και Διαμαντίδης Ν. (2021, Απρίλιος). Τεχνολογία Λογισμικού Β' Έκδοση, εκδόσεις Unibooks

Fitzgibbons, L. (2021, Οκτώβριος). *behavior-driven development (BDD).* Ανάκτηση από TechTarget: https://www.techtarget.com/searchsoftwarequality/definition/Behavior-driven-development-BDD

Hurani, M., & Idris, H. (2024). *Investigating the use of LLMs for.* Karlskrona, Sweeden.

Lee, T. B. (2023, Ιούλιος 31). Ανάκτηση από arstechnica.com: https://arstechnica.com/science/2023/07/a-jargon-free-explanation-of-how-ai-large-language-models-work/6/

Lutkevich, B. (2024, Ιούνιος 21). *TechTarget.* Ανάκτηση από https://www.techtarget.com/whatis/feature/12-of-the-best-large-language-models

Ozkaya, I. C. (2023, Οκτώβριος 2). *Application of Large Language Models (LLMs) in Software Engineering: Overblown Hype or Disruptive Change?.* Ανάκτηση από https://doi.org/10.58012/6n1p-pw64.

Pathak, K. (2022, Σεπτέμβριος 12). *Cucumber Best Practices to follow for efficient BDD Testing*. Ανάκτηση από Medium: https://kailash-pathak.medium.com/cucumber-best-practices-to-follow-for-efficient-bdd-testing-b3eb1c7e9757

Pi, W. (2024, Μάϊος 7). *Research Graph.* Ανάκτηση από Medium: https://medium.com/@researchgraph/brief-introduction-to-the-history-of-large-language-models-llms-3c2efa517112

Sanderson, G. (2017, Οκτώβριος 16). *3blue1brown.* Ανάκτηση από 3blue1brown: https://www.3blue1brown.com/lessons/gradient-descent#another-way-to-think-about-the-gradient

Sumrak, J. (2024, Μάρτιος 11). *7 LLM use cases and applications in 2024.* Ανάκτηση από AssemblyAI: https://www.assemblyai.com/blog/llm-use-cases/

Xinyi Hou, Y. Z. (2023). *Large Language Models for Software Engineering: A Systematic Literature Review.* Ανάκτηση από https://ar5iv.labs.arxiv.org/html/2308.10620

Zharovskikh, A. (2023, Ιούνιος 22). *Best applications of large language models*.

Ανδρουτσόπουλος, Ί. (2024, Φεβρουάριος). Τεχνητή Νοημοσύνη και Μεγάλα Γλωσσικά Μοντέλα. *ΟΠΑ News Εφημερίδα Οικονομικού Πανεπιστημίου Αθηνών Τεύχος 51*, σσ. 8-9.

8. Παραρτήματα

8.1 Φάση 1

8.1.1 GPT -3.5 Συνομιλία 1

Περιγραφή:

Σε αυτήν την αρχική συνομιλία με το GPT-3.5, παρείχα στο LLM τη γνώση της περιγραφής/αρχιτεκτονικής του συστήματος και όλα τα features του συστήματος σε ένα μόνο μήνυμα. Στη συνέχεια, του έδωσα οδηγίες να δημιουργήσει τον κώδικα. Αρχικά, το LLM δεν κατάλαβε πλήρως την εργασία και δημιούργησε μόνο τα Step Definitions με κενό σώμα. Μετά από κάποια καθοδήγηση μέσω μηνυμάτων, άρχισε να δημιουργεί λίγο περισσότερο κώδικα, αλλά σε πολλές περιπτώσεις ανέφερε ότι θα έπρεπε να υλοποιήσω τη λογική του συστήματος ως σχόλιο. Αργότερα, όταν του έδωσα οδηγίες να παρέχει τη λογική του συστήματος (όπως DAOs και domain), έκανε αρκετά καλή δουλειά, ειδικά για τα Data Access Objects. Καθώς η συζήτηση προχωρούσε, το LLM κατάλαβε καλύτερα τις οδηγίες και ήταν πιο πρόθυμο να δημιουργήσει κώδικα για τους ορισμούς βημάτων (σύνδεσμος πλήρους συνομιλίας).

8.1.2 GPT -3.5 Συνομιλία 2

Περιγραφή:

Σ' αυτήν τη συνομιλία με το GPT-3.5, παρείχα στο LLM τη γνώση για την περιγραφή/αρχιτεκτονική του συστήματος και όλα τα features του συστήματος σε μια μόνο εντολή. Στη συνέχεια, ζήτησα από το LLM να δημιουργήσει τον κώδικα, αρχικά για τις κλάσεις domain και, αργότερα, για τα DAOs και Services. Με αυτή τη μικρή διαφοροποίηση, το σύστημα άρχισε αμέσως να δημιουργεί κώδικα χωρίς να παρέχει κενά step definitions, σε αντίθεση με την πρώτη συνομιλία όπου δεν δημιούργησε καθόλου κώδικα στην αρχή. Το LLM παρείχε περισσότερες πληροφορίες μετά από κάθε απάντηση και έκανε εξαιρετική δουλειά με την εκτίμηση των κλάσεων Domain και DAO με βάση τα σενάρια. Συνολικά, η εντολή να δημιουργήσει πρώτα τον κώδικα για τις κλάσεις domain και στη συνέχεια να προχωρήσει στη δημιουργία των step definitions είχε θετική επίδραση στην κατανόηση του συστήματος από το LLM (σύνδεσμος πλήρους συνομιλίας).

8.1.3 GPT -3.5 Συνομιλία 3

Περιγραφή:

Σ' αυτήν τη συνομιλία, προσπάθησα να δώσω τις ίδιες πληροφορίες όπως πάντα, παρέχοντας τα features ένα-ένα, κάτι που φαίνεται να βοηθά το LLM να παραμένει επικεντρωμένο σε κάθε συγκεκριμένο feature και να προσπαθεί να μαντέψει τον κώδικα για τα υπόλοιπα χαρακτηριστικά. Ωστόσο, άλλαξα λίγο την αρχική εντολή για να δω αν θα υπήρχαν σημαντικές αλλαγές, δίνοντας μια πιο λεπτομερή εξήγηση της διαδικασίας που ήθελα να ακολουθήσει το LLM. Το LLM παρείχε καλά αποτελέσματα, μάντεψε κάποιες κλάσεις domain σωστά από την αρχή, μαζί με τα DAOs, και άρχισε να υλοποιεί τον κώδικα άμεσα. Αυτό είναι κοινό σε όλες τις συνομιλίες όπου ζήτησα πρώτα τον κώδικα για τις κλάσεις domain. Σε αντίθεση με άλλες συνομιλίες, όπου το LLM δυσκολευόταν να δημιουργήσει τον κώδικα για τα step definitions χωρίς περισσότερες εξηγήσεις, αυτή η προσέγγιση είχε θετικό αντίκτυπο. Συνολικά, τα αποτελέσματα ήταν καλά για όλα τα features. Το LLM μάντεψε σωστά κάποιες κλάσεις και απαιτήθηκε πολύ λίγη επεξήγηση της εντολής για να κατανοήσει τι χρειαζόταν να κάνει (σύνδεσμος πλήρους συνομιλίας).

8.1.4 GPT -3.5 Συνομιλία 4

Περιγραφή:

Σ' αυτήν τη συνομιλία, επιχείρησα να δώσω στο LLM ένα feature τη φορά, ζητώντας στη συνέχεια τη δημιουργία του κώδικα για τις κλάσεις domain και, κατόπιν, για τα step definitions, για κάθε feature ξεχωριστά. Ο στόχος ήταν να διερευνήσω αν η παροχή των πληροφοριών σε επιπλέον βήματα (με την προσθήκη του κώδικα domain) θα βοηθούσε το σύστημα να παρέχει καλύτερο κώδικα με λιγότερες εντολές και λιγότερη καθοδήγηση. Το LLM κατάλαβε καλά την εργασία και δημιούργησε τον κώδικα για τις κλάσεις domain με τα DAOs, όπως ζητήθηκε, και στη συνέχεια ανέπτυξε τον κώδικα για τα step definitions χωρίς επιπλέον βοήθεια. Αυτό σημαίνει ότι χρειάστηκε λιγότερες εντολές για να ολοκληρώσει την εργασία. Επιπλέον, το σύστημα φαίνεται να κατανοεί καλύτερα την persona που δημιούργησα (“George Red”) και για πρώτη φορά χρησιμοποίησε αυτή την persona απευθείας, αντί να τη χρησιμοποιεί ως μεταβλητή {string}, όπως συνέβαινε σε προηγούμενες συνομιλίες. Συνολικά, τα αποτελέσματα ήταν μέτρια και παρόμοια με αυτά άλλων συνομιλιών. Διαπίστωσα ότι η παροχή των features ένα-ένα δεν φαίνεται να βοηθά ιδιαίτερα, καθώς ένα χαρακτηριστικό στο τελευταίο feature μπορεί να περιέχει πολύτιμες πληροφορίες για ένα σενάριο στο πρώτο feature, που το LLM δεν έχει πάντα επίγνωση. Παρόλα

αυτά, οι εντολές είναι πιο σύντομες και το σύστημα φαίνεται να θυμάται πιο εύκολα το τελευταίο feature που δίνεται (σύνδεσμος πλήρους συνομιλίας).

8.1.5 GitHub Copilot Συνομιλία 1

Περιγραφή:

Στην πρώτη μου αλληλεπίδραση με το GitHub Copilot, δεν χρειάστηκε να παρέχω τα features του συστήματος, καθώς το σύστημα μπορούσε να διαβάσει αυτόματα τα αρχεία από τον editor. Αντίθετα, παρείχα στο LLM την αρχιτεκτονική και τη γενική γνώση του συστήματος καθώς και τα features του. Αρχικά, το LLM παρήγαγε μόνο τα Step Definitions με κενό σώμα, αλλά μετά από μερικά μηνύματα, κατανόησε την εργασία και άρχισε να δημιουργεί κώδικα με ευκολία. Για πρώτη φορά, με την περιορισμένη γνώση που του είχε δοθεί, το LLM κατάφερε να κατανοήσει τη σημασία των «περσόνων» (π.χ. George Red) και χρησιμοποίησε σωστά τις μεταβλητές Harry Potter και Moby Dick. Ωστόσο, όπως και το ChatGPT, το LLM χρησιμοποίησε συχνά assertions στα βήματα GIVEN, κάτι που δεν είναι η καλύτερη πρακτική για την γραφή των Step Definitions, καθώς τα βήματα GIVEN πρέπει να δημιουργούν το περιβάλλον και τις μεταβλητές για την επαλήθευση του σεναρίου. Συνολικά, το LLM αντελήφθη αμέσως τις εντολές και χρειάστηκε λίγα μηνύματα για να δημιουργήσει τον κώδικα, σε αντίθεση με το ChatGPT. Ο παραγόμενος κώδικας δεν ήταν τέλειος, αλλά ήταν αρκετά καλός σε πολλά σημεία. Το LLM έκανε καλή χρήση των Service κλάσεων, αλλά όχι των Data Access Objects, και παρέλειψε κάποια σημαντικά Step Definitions λόγω της πολυπλοκότητάς τους και της έλλειψης προϋπάρχουσας γνώσης που του είχε δοθεί (σύνδεσμος πλήρους συνομιλίας).

8.1.6 GitHub Copilot Συνομιλία 2

Σε αυτήν τη συνομιλία, παρείχα τις ίδιες πληροφορίες όπως και στις προηγούμενες συνομιλίες, με ελαφρές διαφορές στην αρχική εντολή. Ωστόσο, αντί να χρησιμοποιήσω το σύστημα αναφοράς του GitHub Copilot, συμπεριέλαβα τα features του συστήματος ως μήνυμα. Αρχικά, το LLM άρχισε να δημιουργεί τις κλάσεις domain, DAOs και Services που χρειάζονταν, χωρίς να του το ζητήσω ρητά, κάτι που δεν είχε κάνει ποτέ με το ChatGPT. Αυτό είναι θετικό, καθώς δείχνει ότι το LLM σκέφτεται προληπτικά για τις κλάσεις domain πρώτα, πριν προχωρήσει στη δημιουργία του κώδικα που ζήτησα αργότερα. Οι κλάσεις domain ήταν ικανοποιητικές. Στη συνέχεια, το LLM άρχισε να δημιουργεί τον κώδικα των step definitions από την αρχή, χωρίς να χρειαστούν πολλές εντολές. Ο κώδικας ήταν παρόμοιος με τις άλλες περιπτώσεις, αλλά το σύστημα χρησιμοποίησε

περισσότερα Services, κάτι που είναι πολύ καλό, και κατάλαβε ότι έπρεπε να χρησιμοποιεί DAOs (χρησιμοποίησε κλάσεις BorrowerService που χρησιμοποιούν DAOs). Συνολικά, το LLM ήταν πολύ καλό στην κατανόηση του τι έπρεπε να κάνει. Η δημιουργία του κώδικα domain πρώτα φαίνεται να βοηθά στην πιο αποτελεσματική δημιουργία του κώδικα. Θα δοκιμαστεί περαιτέρω για να επιβεβαιωθεί αν είναι καλύτερο να χρησιμοποιείται το σύστημα αναφοράς ή να παρέχονται τα σενάρια μέσω εντολών. Ωστόσο, με βάση αυτή τη συνομιλία, φαίνεται ότι οι εντολές είναι η καλύτερη επιλογή (σύνδεσμος πλήρους συνομιλίας).

8.1.7 GitHub Copilot Συνομιλία 3

Περιγραφή:

Αυτή η συνομιλία ήταν ένα πείραμα για να διαπιστώσω αν το LLM θα μου παρέχει ξανά τις κλάσεις Domain ως αρχικό σημείο, χωρίς να του το ζητήσω, κάτι που τελικά έκανε. Στη συνέχεια, συνέχισα τη συνομιλία για να αποκτήσω τον πλήρη κώδικα των step definitions. Τα αποτελέσματα ήταν συνολικά καλά και αρκετά παρόμοια με τα αποτελέσματα των προηγούμενων συνομιλιών, με βάση τις γνώσεις που παρείχα στο LLM. Το σύστημα άρχισε να δημιουργεί κώδικα αμέσως, απαιτώντας πολύ λίγες εντολές (σύνδεσμος πλήρους συνομιλίας).

8.1.8 GitHub Copilot Συνομιλία 4

Περιγραφή:

Σ' αυτήν τη συνομιλία με το GitHub Copilot, παρείχα στο LLM την αρχιτεκτονική και τη γενική γνώση του συστήματος καθώς και τα Features του συστήματος σε μία εντολή, όπως έκανα και σε προηγούμενες συνομιλίες. Ο στόχος ήταν να δω αν το LLM θα δημιουργήσει τις ίδιες κλάσεις domain όπως στην αρχή, όταν είχα δώσει τα Features μέσω του συστήματος αναφοράς του GitHub. Τα αποτελέσματα ήταν αρκετά παρόμοια, γεγονός που έκανε τη συνομιλία λιγότερο ενδιαφέρουσα (σύνδεσμος πλήρους συζήτησης).

8.1.9 GPT -4 Συνομιλία 1

Περιγραφή:

Στην πρώτη συνομιλία με το GPT-4, παρείχα στο LLM όλα τα features του συστήματος καθώς και την αρχιτεκτονική του ταυτόχρονα. Το LLM χρειάστηκε μερικές εντολές για να κατανοήσει ακριβώς τι ήθελα, και έπρεπε να επαναλάβω πολλές φορές τις εντολές σχετικά με τη δημιουργία όλου του κώδικα και όλων των step definitions για τα features. Συνολικά, τα αποτελέσματα δεν

ήταν πολύ καλά. Ο κώδικας ήταν συχνά πολύ απλός, και το LLM απαιτούσε πολλές εντολές για να κατανοήσει τι έπρεπε να δημιουργήσει. Επιπλέον, δεν χρησιμοποίησε αυτόνομα τα DAOs και χρειαζόταν σαφείς οδηγίες για να τα ενσωματώσει (σύνδεσμος πλήρους συνομιλίας).

8.1.10 GPT -4 Συνομιλία 2

Περιγραφή:

Σε αυτήν τη συνομιλία, παρείχα στο LLM όλα τα features του συστήματος καθώς και την αρχιτεκτονική του, και ζήτησα επίσης να δημιουργήσει αρχικά τον κώδικα για τα Domain, DAOs και Services, πριν προχωρήσει στη δημιουργία κώδικα για τα step definitions. Το LLM κατάλαβε την εργασία και δημιούργησε τον κώδικα όπως ζητήθηκε. Ωστόσο, χρειάστηκε επαναλαμβανόμενες εντολές για να παράγει όλα τα step definitions για τα features, χρησιμοποιώντας την αρχιτεκτονική που παρείχα από την αρχή. Συνολικά, τα αποτελέσματα δεν ήταν κακά για μερικά από τα step definitions, αλλά ήταν πολύ απλά και δεν ανταγωνίζονταν το επίπεδο του πραγματικού συστήματος για πολλά άλλα step definitions (σύνδεσμος πλήρους συνομιλίας).

8.1.11 GPT -4 Συνομιλία 3

Περιγραφή:

Σε αυτήν τη συνομιλία, παρείχα στο LLM τα features του συστήματος ένα προς ένα, αντί να τα δώσω όλα μαζί, καθώς και την αρχιτεκτονική. Αυτή η προσέγγιση είχε ως αποτέλεσμα το LLM να θυμάται καλύτερα τις εντολές, καθώς κάθε feature δινόταν ξεχωριστά. Ωστόσο, το LLM δεν είχε γνώση των μεταγενέστερων features στις πρώτες υλοποιήσεις των step definitions. Συνολικά, το LLM κατάλαβε αρκετά καλά την εργασία, αλλά χρειάστηκε και πάλι επαναλαμβανόμενες εντολές για να δημιουργήσει τα πλήρη step definitions για τα features, όπως συνέβη και σε όλες τις άλλες περιπτώσεις (σύνδεσμος πλήρους συνομιλίας).

8.1.12 GPT -4 Συνομιλία 4

Περιγραφή:

Σε αυτήν τη συνομιλία, παρείχα στο LLM όλα τα features του συστήματος ένα προς ένα, μαζί με την αρχιτεκτονική. Πριν από κάθε feature, ζήτησα από το LLM να δημιουργήσει πρώτα τον κώδικα για τις κλάσεις domain και στη συνέχεια να προχωρήσει στη δημιουργία του κώδικα για τα step definitions. Όπως και σε όλες τις άλλες συνομιλίες, το LLM χρειάστηκε

επαναλαμβανόμενες εντολές για να δημιουργήσει πλήρως τα step definitions. Τα αποτελέσματα ήταν καλά σε ορισμένες περιπτώσεις, αλλά όχι εξαιρετικά. Το LLM φάνηκε να κατανοεί καλύτερα την εργασία, αν και της έλειπε κρίσιμη γνώση των μεταγενέστερων features, κάτι που θα μπορούσε να έχει βοηθήσει στην υλοποίηση των προηγούμενων features (σύνδεσμος πλήρους συνομιλίας).

8.1.13 GPT -4o Συνομιλία 1

Περιγραφή:

Στην πρώτη συνομιλία με το GPT-4o, παρείχα τις ίδιες πληροφορίες που είχα δώσει σε άλλες συνομιλίες στη φάση 1 με διάφορα LLMs (features + σύντομη περιγραφή του συστήματος), αλλά χωρίς να δώσω επιπλέον οδηγίες για τη δημιουργία του κώδικα Domain ή άλλων στοιχείων, όπως έκανα σε άλλες περιπτώσεις. Τα αποτελέσματα ήταν αρκετά ικανοποιητικά για την ποσότητα των πληροφοριών που παρείχα. Το LLM κατανόησε άμεσα την εργασία και δημιούργησε το πρώτο feature με ευκολία. Ειδικότερα, το LLM δεν παρέλειψε κανένα σενάριο και ανέπτυξε ένα χαρακτηριστικό που δεν είχα παρατηρήσει σε προηγούμενες συνομιλίες: για κάθε απάντηση βασισμένη σε διαφορετικό feature, θυμόταν την αρχική εντολή που ανέφερε την αρχιτεκτονική (Domain, DAOs, Services) και προσαρμόστηκε αναλόγως. Δημιούργησε τον κώδικα για DAOs, Services και Domain, κάνοντας αναγκαίες αλλαγές με βάση τις νέες πληροφορίες. Επιπλέον, για πρώτη φορά, το LLM παρείχε και τις εξαρτήσεις για την προσθήκη του Cucumber στο σύστημα, κάτι που δεν είχε γίνει σε προηγούμενες συνομιλίες. Συνολικά, τα αποτελέσματα ήταν αρκετά καλά για μια πρώτη συνομιλία. Το LLM χρησιμοποίησε assertions στα βήματα GIVEN, κάτι που ίσως δεν είναι ιδανικό, αλλά κατανόησε γρήγορα τι απαιτείτο και δημιούργησε τον απαιτούμενο κώδικα με πολύ λίγες καθοδηγήσεις, χρησιμοποιώντας αποτελεσματικά τα DAOs, τις κλάσεις Domain και τα Services (σύνδεσμος πλήρους συνομιλίας).

8.1.14 GPT -4o Συνομιλία 2

Περιγραφή:

Στη συνομιλία αυτή, παρείχα ακριβώς τις ίδιες πληροφορίες όπως στη Συνομιλία 1, αλλά πρόσθεσα την οδηγία να δημιουργήσει πρώτα τον κώδικα για τις κλάσεις Domain, Services και DAOs. Η προσέγγιση αυτή είχε ως στόχο να δούμε αν θα επηρεάσει τις απαντήσεις του LLM. Το GPT-4o κατάλαβε εξαιρετικά καλά την εργασία από την αρχή και, όπως και στην πρώτη

συνομιλία, παρείχε τις κλάσεις Domain, DAOs και Services με λεπτομέρεια. Ενώ μερικά χαρακτηριστικά στις κλάσεις Domain ήταν σωστά και άλλα όχι, το LLM περιέγραψε όλες τις κλάσεις και τις μεθόδους λεπτομερώς. Μετά τη δημιουργία των κλάσεων, το LLM δημιούργησε τα features με ευκολία και χωρίς πολλές επιπλέον καθοδηγήσεις. Συνολικά, τα αποτελέσματα ήταν παρόμοια με εκείνα της Συνομιλίας 1, με το LLM να παρέχει επίσης την αρχιτεκτονική που θα έπρεπε να ακολουθήσετε στη δομή των αρχείων σας. Η πρόσθετη οδηγία για τη δημιουργία των κλάσεων Domain/Services/DAOs πρώτα δεν φαίνεται να είχε σημαντική επίδραση στα αποτελέσματα. Το GPT-4o συνέχισε να παράγει αποτελέσματα υψηλής ποιότητας, χρησιμοποιώντας αποτελεσματικά τα DAOs, τις κλάσεις Domain και τα Services όπως απαιτείτο. Οι απαντήσεις ήταν και πάλι χρήσιμες και σωστές, δείχνοντας ότι το LLM διαχειριζόταν καλά την αρχιτεκτονική και τις απαιτήσεις του συστήματος (σύνδεσμος πλήρους συνομιλίας).

8.1.15 GPT -4o Συνομιλία 3

Περιγραφή:

Στη συνομιλία αυτή, χρησιμοποιήθηκε η προσέγγιση της παροχής των features ένα προς ένα, όπως έγινε με τα άλλα LLM. Αυτή η τεχνική στόχευε στο να δούμε αν το LLM θα επικεντρωνόταν περισσότερο σε κάθε μεμονωμένο feature και θα βελτίωνε την ποιότητα του κώδικα του. Συνολικά, τα αποτελέσματα ήταν σχεδόν τα ίδια με αυτά που είχατε πάρει πριν ή, σε κάποιες περιπτώσεις, ήταν ακόμα χειρότερα. Το LLM φάνηκε να κατανοεί γρήγορα τα features και να δημιουργεί τις απαντήσεις αμέσως, αλλά η προσέγγιση της παροχής των features ένα προς ένα δεν είχε θετική επίδραση στην ποιότητα του κώδικα που παρήχθη. Η παροχή όλων των πληροφοριών από την αρχή θα μπορούσε να είχε βελτιώσει την ποιότητα του κώδικα, δεδομένου ότι το LLM θα είχε πλήρη εικόνα του συστήματος από την αρχή. Η τεχνική της παροχής features ένα προς ένα δεν έφερε την επιθυμητή βελτίωση και συνεπώς δεν θα χρησιμοποιηθεί σε μελλοντικές συνομιλίες, καθώς δεν είχε θετική επίδραση στην παραγωγή καλύτερου κώδικα (σύνδεσμος πλήρους συνομιλίας).

8.2 Φάση 2

8.2.1 GPT -3.5 Συνομιλία 1

Περιγραφή:

Σ' αυτήν τη συνομιλία της φάσης 2, παρέδωσα στο LLM τα features του συστήματος σε μορφή Gherkin και τα ονόματα των κλάσεων domain, ζητώντας να χρησιμοποιήσει Data Access Objects (DAOs) και Services (χωρίς να παρέχω τις κλάσεις για αυτά). Η εντολή περιλάμβανε όλα τα features σε ένα μήνυμα, και ζήτησα από το LLM να παρέχει πρώτα τον κώδικα για τις κλάσεις domain, DAOs και Services πριν δημιουργήσει τον κώδικα για τα step definitions. Τα αποτελέσματα ήταν εντυπωσιακά. Το LLM κατάφερε να μαντέψει τον κώδικα για τις κλάσεις με αρκετή ακρίβεια. Συγκεκριμένα, η πρότασή του για την κλάση `ItemState` ως Enumeration ήταν ακριβής, όπως αποδείχθηκε από τον πραγματικό κώδικα. Για τα DAOs, το LLM δημιούργησε περισσότερες κλάσεις από όσες ήταν απαραίτητες, αλλά έδειξε καλή κατανόηση των συναρτήσεων που απαιτούνται. Επίσης, έκανε καλή δουλειά με ορισμένα Services, όπως το `NotificationService` και το `LoanService`, αν και χρησιμοποίησε και επιπλέον υπηρεσίες. Συνολικά, ο παραγόμενος κώδικας ήταν απλός και εκτελέσιμος, αν και δεν ταίριαζε πάντα με το πραγματικό back-end του συστήματος. Η εντολή να δημιουργήσει τον κώδικα για τις κλάσεις domain, DAOs και Services μαζί φαίνεται να βοήθησε το LLM να ενσωματώσει αυτά τα στοιχεία πιο αποτελεσματικά στις λύσεις του. Παρόλα αυτά, το LLM άφησε πολλές step definitions κενές ή πρόσφερε σχόλια για να υλοποιήσω εγώ τη λογική, κάτι που υποδεικνύει ότι δεν είχε πλήρη κατανόηση των απαιτήσεων από την αρχή (σύνδεσμος πλήρους συνομιλίας).

8.2.2 GPT -3.5 Συνομιλία 2

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρέδωσα τις ίδιες πληροφορίες όπως στη Φάση 2, Συνομιλία 1 (features και κλάσεις domain), αλλά δεν ζήτησα από το LLM να δημιουργήσει ή να μαντέψει πρώτα τον κώδικα για τις κλάσεις domain. Αντίθετα, ζήτησα να δημιουργήσει απευθείας τα step definitions. Αρχικά, το LLM δεν δημιούργησε πραγματικό κώδικα και παρέδωσε μόνο τα ονόματα των κλάσεων των step definitions. Με κάποια επιπλέον βοήθεια και εντολές, το LLM παρείχε τελικά κώδικα για τα step definitions, αλλά χρησιμοποιούσε και σχόλια για να εξηγήσει τι έπρεπε να γίνει. Ένα σημαντικό πρόβλημα ήταν ότι το LLM συγκέντρωσε όλη τη λογική του συστήματος σε μια κλάση με το όνομα `libraryService`, γεγονός που δεν ήταν η καλύτερη πρακτική κωδικοποίησης. Αυτό δείχνει ότι το LLM προτίμησε να χρησιμοποιήσει μια ενιαία κλάση για να αποφύγει την πολυπλοκότητα του συστήματος, κάτι που δεν ακολουθεί τις συνήθεις καλές πρακτικές για τον διαχωρισμό ευθυνών. Συγκρίνοντας με τη Φάση 2, Συνομιλία 1, όπου ζητήθηκε

πρώτα ο κώδικας για τις κλάσεις domain και στη συνέχεια για τα step definitions, η προσέγγιση αυτή είχε καλύτερα αποτελέσματα. Στη συγκεκριμένη συνομιλία, το LLM κατάφερε να μαντέψει σωστά πολλές κλάσεις domain και να δημιουργήσει πιο συνεπή κώδικα, όπως την κλάση Enum για το ItemState. Τελικά, όταν ζήτησα από το LLM να παρέχει τον κώδικα για τις κλάσεις domain μετά την ολοκλήρωση του κώδικα για όλα τα step definitions, τα αποτελέσματα ήταν λιγότερο ικανοποιητικά σε σύγκριση με την πρώτη φάση, όπου το LLM είχε καλύτερη κατανόηση των απαιτήσεων και δημιούργησε πιο ακριβείς κλάσεις domain. Αυτό υποδηλώνει ότι η ενσωμάτωση των κλάσεων domain στην αρχή της διαδικασίας μπορεί να βοηθήσει το LLM να κατανοήσει και να δημιουργήσει πιο ακριβή και οργανωμένο κώδικα για τα step definitions (σύνδεσμος πλήρους συνομιλίας).

8.2.3 GPT -3.5 Συνομιλία 3

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρείχα στο LLM τα features και τις κλάσεις domain του συστήματος, αλλά παρουσίασα τα features ένα-ένα αντί να τα δώσω όλα μαζί σε μία εντολή. Ζήτησα από το LLM να δημιουργήσει τον κώδικα για τις κλάσεις domain και, στη συνέχεια, τον κώδικα για τα step definitions για κάθε feature ξεχωριστά. Το LLM φαινόταν να κατανοεί πολύ καλά την εργασία και άρχισε να δημιουργεί κώδικα από το πρώτο μήνυμα. Τα αποτελέσματα ήταν ικανοποιητικά, αλλά το κύριο πρόβλημα ήταν ότι, επειδή δεν είχε όλες τις πληροφορίες για τα features από την αρχή, δεν δημιούργησε κάποιες κλάσεις που ήταν προφανές ότι έπρεπε να δημιουργηθούν αν είχε διαβάσει το τελευταίο feature. Αυτές οι κλάσεις θα ήταν πολύ χρήσιμες για το πρώτο feature, με αποτέλεσμα ο κώδικας να μην ήταν τέλειος. Συνολικά, ωστόσο, η προσέγγιση της παροχής μικρών, καθοδηγητικών μηνυμάτων στο LLM, αντί να της ζητάς να κάνει τα πάντα με μία εντολή, φαίνεται να είναι μερικές φορές λίγο καλύτερη (σύνδεσμος πλήρους συνομιλίας).

8.2.4 GitHub Copilot Συνομιλία 1

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρείχα στο LLM τα Features του συστήματος ως εντολή και τα ονόματα των κλάσεων Domain που έπρεπε να χρησιμοποιήσει. Αυτή τη φορά, το LLM δεν δημιούργησε τον κώδικα για τις κλάσεις Domain, όπως είχε γίνει στη Φάση 1, πιθανώς επειδή ήδη είχε τα ονόματα των κλάσεων Domain. Αντίθετα, άρχισε αμέσως να δημιουργεί κώδικα (χωρίς να ζητήσω

πρώτα τη δημιουργία των κλάσεων Domain). Ο κώδικας που παρήχθη ήταν αρκετά απλός σε πολλές περιπτώσεις (π.χ., assertions στα βήματα GIVEN κ.λπ.). Το LLM χρησιμοποίησε καλά τις κλάσεις service, προσθέτοντας μερικές επιπλέον, αλλά το αποτέλεσμα ήταν παρόμοιο με το GPT-3.5. Ωστόσο, το LLM παρέλειψε εντελώς τη χρήση των DAOs και χρησιμοποίησε μόνο τις κλάσεις domain που παρείχα, παραλείποντας πολλές κλάσεις όπως η Book χωρίς να τις ενσωματώσει καθόλου. Συνολικά, το LLM δημιούργησε κώδικα πολύ εύκολα και δεν χρειάστηκε σχεδόν καθόλου εντολές, αλλά ο κώδικας ήταν απλός. Αυτό δείχνει ότι η παροχή μόνο των ονομάτων των κλάσεων domain δεν βοηθάει πραγματικά το Copilot να δημιουργήσει πιο σύνθετο κώδικα (το ίδιο είχε συμβεί και με το GPT-3.5). Ωστόσο, το LLM κατανόησε πολύ καλά την εργασία από την αρχή (σύνδεσμος πλήρους συνομιλίας).

8.2.5 GitHub Copilot Συνομιλία 2

Περιγραφή:

Σ' αυτήν τη συνομιλία, έδωσα στο Copilot τις ακριβώς ίδιες πληροφορίες όπως στη συνομιλία 1, αλλά προσπάθησα να παρέχω τα features ως «αναφορές» (χρησιμοποιώντας το σύστημα αναφοράς του GitHub Copilot, όπου παρέχεις τα αρχεία των features). Τα αποτελέσματα ήταν σχεδόν τα ίδια, αν και το LLM δεν δημιούργησε τον κώδικα για μερικά σενάρια, ακόμα και όταν το πίεσα να το κάνει, και μου έδωσε σχόλια για να υλοποιήσω τη λογική. Από εδώ και πέρα, θα χρησιμοποιώ την τεχνική της εντολής για την παροχή πληροφοριών, καθώς φαίνεται να είναι πιο αποτελεσματική (σύνδεσμος πλήρους συνομιλίας).

8.2.6 GitHub Copilot Συνομιλία 3

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρέδωσα στο Copilot τις ίδιες πληροφορίες όπως και στις προηγούμενες συνομιλίες, αλλά ζήτησα από το LLM να δημιουργήσει πρώτα τις κλάσεις domain, όπως έκανα στη φάση 2 του GPT-3.5. Τα αποτελέσματα για τις κλάσεις domain ήταν τα καλύτερα από όλες τις συνομιλίες της φάσης 2 (και των δύο LLMs). Το LLM κατάλαβε πολύ καλά τις ιδιότητες που έπρεπε να έχουν οι κλάσεις, για παράδειγμα, πρόσθεσε το Book ως χαρακτηριστικό στο Item και μάντεψε σωστά τον Enum για το ItemState, καθώς και μερικά άλλες εξαιρετικές ιδιότητες. Ωστόσο, ο κώδικας για τα step definitions ήταν αρκετά παρόμοιος με τις προηγούμενες συνομιλίες και το LLM ξέχασε ξανά να χρησιμοποιήσει DAOs. Παρά αυτά τα σημεία, το LLM κατανοούσε

πολύ καλά την persona George Red και τα ονόματα των items όπως Harry Potter, Animal Kingdom κ.λπ., όπως και σε όλες τις άλλες συνομιλίες (σύνδεσμος πλήρους συνομιλίας).

8.2.7 GitHub Copilot Συνομιλία 4

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρείχα στο LLM τις ίδιες πληροφορίες όπως στη Φάση 2, Συνομιλία 3, αλλά ζήτησα επίσης τη δημιουργία του κώδικα για τις κλάσεις domain, καθώς και για τα DAOs και τα Services, προκειμένου το σύστημα να χρησιμοποιεί πραγματικά DAOs. Όπως αναμενόταν, το LLM έκανε εξαιρετική δουλειά στη δημιουργία των κλάσεων domain, καθώς και των DAOs και Services. Ήταν η πρώτη φορά που το LLM δημιούργησε ένα interface για τα DAOs. Στη συνέχεια, όταν ζήτησα από το LLM να δημιουργήσει τον κώδικα, διαπίστωσα ότι ήταν σχεδόν αντιγραφή-επικόλληση από τις προηγούμενες συνομιλίες 1, 2 και 3. Συνολικά, η οδηγία προς το LLM να δημιουργήσει επίσης τον κώδικα για τα DAOs φαίνεται να βοηθά στην αποτελεσματική χρήση τους σε επόμενα βήματα (σύνδεσμος πλήρους συνομιλίας).

8.2.8 GitHub Copilot Συνομιλία 5

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρείχα στο LLM τις ίδιες πληροφορίες όπως και σε όλες τις άλλες συνομιλίες, αλλά αυτή τη φορά παρέδωσα τα features ένα-ένα, αντί να τα δώσω όλα από την αρχή. Για κάθε feature, ζήτησα από το σύστημα να παρέχει τον κώδικα για τις κλάσεις domain, DAOs και Services. Το LLM κατάλαβε αρκετά καλά την εργασία, έκανε ακριβώς ό,τι ζήτησα, και για κάθε νέο feature δημιούργησε ξανά τις κλάσεις domain, DAOs και Services, ενισχυμένες με τις νέες ιδιότητες που προέκυπταν από το feature. Επίσης, παρείχε αρκετά καλό κώδικα για τα Services και για τα τελευταία features, όπου το LLM είχε τις περισσότερες πληροφορίες, δημιούργησε πολύ καλό κώδικα. Το μόνο πρόβλημα ήταν ότι αν είχα παρέχει όλα τα features από την αρχή, το LLM θα μπορούσε να είχε χρησιμοποιήσει μερικές από τις νέες ιδιότητες στις πρώτες απαντήσεις για τα step definitions. Ωστόσο, η στρατηγική της παροχής των features ένα-ένα δεν ήταν κακή και η προσέγγιση αυτή ενδέχεται να έχει τα δικά της πλεονεκτήματα (σύνδεσμος πλήρους συνομιλίας).

8.2.9 GPT -4 Συνομιλία 1

Περιγραφή:

Σε αυτήν τη συνομιλία, παρείχα στο LLM όλα τα features του συστήματος, τα ονόματα των κλάσεων domain, καθώς και την αρχιτεκτονική και γενικές πληροφορίες για το σύστημα. Το LLM κατάλαβε καλά την εργασία με πολύ λίγες εντολές, αλλά χρειάστηκε λίγη επανάληψη για να παρέχει τον πλήρη κώδικα υλοποίησης των step definitions. Συνολικά, το LLM ακολούθησε την αρχιτεκτονική που του παρείχα και δημιούργησε καλές απαντήσεις σε ορισμένα σενάρια. Ωστόσο, ο κώδικας ήταν επίσης υπερβολικά απλουστευμένος σε πολλές άλλες περιπτώσεις (σύνδεσμος πλήρους συνομιλίας).

8.2.10 GPT -4 Συνομιλία 2

Περιγραφή:

Σε αυτή τη συνομιλία, παρείχα στο LLM όλα τα features του συστήματος, τα ονόματα των κλάσεων domain που πρέπει να χρησιμοποιήσει, καθώς και την αρχιτεκτονική και γενικές πληροφορίες για το σύστημα. Η διαφορά σε αυτή την περίπτωση ήταν ότι ζήτησα πρώτα από το LLM να σκεφτεί και να δημιουργήσει τον κώδικα για τις κλάσεις domain. Το LLM εκτίμησε αρκετά καλά τις σχέσεις μεταξύ των κλάσεων και μερικά features. Ξεκίνησε να δημιουργεί κώδικα από την αρχή, πράγμα που δείχνει ότι κατανοούσε πολύ καλά τι του παρείχα. Ωστόσο, χρειάστηκε και πάλι κάποια επανάληψη στις εντολές για να δημιουργήσει όλα τα step definitions, παραλείποντας μερικά. Συνολικά, δημιούργησε αρκετά περίπλοκο κώδικα, ο οποίος ήταν καλός σε ορισμένες περιπτώσεις αλλά κακός σε άλλες, επειδή ήταν υπερβολικά περίπλοκος για να κατανοηθεί εύκολα το step definition (σύνδεσμος πλήρους συνομιλίας).

8.2.11 GPT -4 Συνομιλία 3

Περιγραφή:

Σε αυτήν τη συνομιλία, παρείχα στο LLM όλα τα features του συστήματος ένα προς ένα, τα ονόματα των κλάσεων domain που πρέπει να χρησιμοποιήσει, καθώς και την αρχιτεκτονική και γενικές πληροφορίες για το σύστημα. Το LLM κατάλαβε αρκετά καλά την εργασία, αν και χρειάστηκε και πάλι μερικές επαναλήψεις για να δημιουργήσει τα step definitions για όλα τα features, και παρήγαγε χρήσιμο κώδικα σε πολλές περιπτώσεις. Ωστόσο, το πρόβλημα με αυτήν την τεχνική είναι ότι θα ήταν χρήσιμο αν το LLM γνώριζε νωρίτερα κάποια features που δόθηκαν αργότερα, καθώς αυτό θα μπορούσε να βοηθήσει στη βελτίωση των απαντήσεών του και της

υλοποίησης των χαρακτηριστικών. Συνολικά, δεν ήταν μια κακή συνομιλία (σύνδεσμος πλήρους συνομιλίας).

8.2.12 GPT -4 Συνομιλία 4

Περιγραφή:

Σε αυτή τη συνομιλία, παρείχα στο LLM όλα τα features του συστήματος ένα προς ένα, μαζί με τα ονόματα των κλάσεων domain που πρέπει να χρησιμοποιήσει, την αρχιτεκτονική και γενικές πληροφορίες για το σύστημα. Η διαφορά αυτή τη φορά ήταν ότι, μετά από κάθε feature που δόθηκε, ζήτησα από το LLM να δημιουργήσει τον κώδικα για τις κλάσεις domain, DAOs και services. Το LLM κατάλαβε πολύ καλά αυτό το βήμα και δεν το ξέχασε, δημιουργώντας κάθε feature με την ίδια διαδικασία. Ωστόσο, χρειάστηκε κάποια επανάληψη και υπενθύμιση για να ολοκληρώσει όλα τα step definitions για κάθε feature. Συνολικά, χρησιμοποίησε αρκετά καλά την αρχιτεκτονική και μάντεψε πολλές ιδιότητες και σχέσεις μεταξύ των κλάσεων. Παρά την καλή χρήση της αρχιτεκτονικής, οι step definitions δεν ήταν όλες εξαιρετικές. Μερικές ήταν πολύ απλοϊκές, ενώ κάποιες άλλες ήταν αποδεκτές (σύνδεσμος πλήρους συνομιλίας).

8.2.13 GPT -4o Συνομιλία 1

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM τα features του συστήματος, την περιγραφή του, καθώς και τα ονόματα των κλάσεων Domain που χρειάζονταν. Το LLM κατανόησε άριστα την εργασία, όπως και στην προηγούμενη φάση, και άρχισε να δημιουργεί τον κώδικα για τις κλάσεις Domain (χωρίς προηγούμενη οδηγία για τη δημιουργία του κώδικα Domain), με χαρακτηριστικά που μάντεψε εξαιρετικά καλά, σχεδόν ακριβώς όπως τα πραγματικά χαρακτηριστικά των κλάσεων Domain. Επίσης, δημιούργησε αμέσως τον κώδικα για τα features με πολύ λίγες οδηγίες, καθώς είχε σαφή κατανόηση του τι έπρεπε να κάνει. Συνολικά, ο κώδικας ήταν πολύ καλύτερος σε σύγκριση με την προηγούμενη φάση, με το LLM να παράγει κώδικα με πολλές λεπτομέρειες και καλή κατανόηση του συστήματος και των Step Definitions. Συμπερασματικά, η πρόσθετη γνώση των ονομάτων των κλάσεων Domain είχε σημαντική θετική επίδραση στο αποτέλεσμα (σύνδεσμος πλήρους συνομιλίας).

8.2.14 GPT -4o Συνομιλία 2

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM τις ίδιες ακριβώς πληροφορίες όπως στη Συνομιλία 1, αλλά ήθελα να του ζητήσω να δημιουργήσει πρώτα τον κώδικα για το Domain (κάτι που είχε κάνει αυτόματα σε όλες τις προηγούμενες περιπτώσεις). Παρ' όλα αυτά, το LLM δημιούργησε τον κώδικα Domain χωρίς να του δώσω τα features, χρησιμοποιώντας μόνο τα ονόματα των κλάσεων Domain. Στη συνέχεια, του παρείχα τα features και ζήτησα να δημιουργήσει ξανά τον επικαιροποιημένο κώδικα Domain. Συνολικά, ο κώδικας των κλάσεων Domain ήταν κοντά στον κώδικα του συστήματος, αλλά περιείχε πολλά επιπλέον χαρακτηριστικά. Το LLM χρησιμοποίησε μόνο ένα Service, συγχωνεύοντας όλες τις κλάσεις Services σε αυτό (αντί να δημιουργήσει περισσότερα Services, όπως θα ήταν πιο σωστό). Εντυπωσιακά, για πρώτη φορά το LLM δημιούργησε κώδικα για κάθε Step Definition όλων των features ταυτόχρονα (σε μία μόνο απάντηση) χωρίς να παραλείψει τίποτα, κάτι που είναι εξαιρετικό και πολύ αποδοτικό χρονικά. Συνολικά, ο κώδικας που παρείχε δεν ήταν καλύτερος από τον κώδικα της Συνομιλίας 1, αλλά περιλάμβανε πολλές λεπτομέρειες στην υλοποίησή του και έδειξε βαθιά κατανόηση του κώδικα και των απαιτήσεων. Ενδέχεται, με περισσότερες πληροφορίες, να παραχθούν ακόμα καλύτερα αποτελέσματα (σύνδεσμος πλήρους συνομιλίας).

8.2.15 GPT -4o Συνομιλία 3

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα τις ίδιες πληροφορίες όπως στη Συνομιλία 2 και ζήτησα από το LLM να δημιουργήσει πρώτα τον κώδικα για το Domain, κάτι που έκανε πολύ εύκολα. Το LLM κατάλαβε εξαιρετικά καλά την εργασία και δημιούργησε τον κώδικα για το Domain, Services και DAOs, και στη συνέχεια άρχισε να δημιουργεί τα Step Definitions για ΟΛΑ τα features με ΜΟΝΟ 2 μηνύματα. Αυτό δείχνει ότι το LLM διαθέτει ισχυρή μνήμη για να θυμάται τα Step Definitions κάθε feature. Τα αποτελέσματα του κώδικα ήταν συνολικά τα ίδια με τις προηγούμενες συνομιλίες, με πολλές λεπτομέρειες, αν και σε πολλές περιπτώσεις δεν ήταν τόσο κοντά στον πραγματικό κώδικα (σύνδεσμος πλήρους συνομιλίας).

8.3 Φάση 3

8.3.1 GPT -3.5 Συνομιλία 1

Περιγραφή:

Σ' αυτή την πρώτη συνομιλία της φάσης 3, παρείχα στο LLM τα features του συστήματος και τις κλάσεις domain μαζί με τις ιδιότητές τους. Ζήτησα από το LLM να μου δώσει τον κώδικα υλοποίησης των step definitions. Αρχικά, το LLM δεν παρείχε αμέσως τον κώδικα, αλλά μου έδωσε μόνο τα ονόματα των συναρτήσεων για τα step definitions με σχόλιο που ανέφερε ότι έπρεπε να υλοποιήσω τον κώδικα. Μετά από μερικές εντολές, άρχισε να δημιουργεί κώδικα. Συνολικά, το σύστημα χρησιμοποίησε με επιτυχία DAOs για την αποθήκευση, διαγραφή και πρόσβαση σε αντικείμενα (σε άλλες συνομιλίες, χρησιμοποιούσε Services για την υλοποίηση αυτής της λογικής). Ο κώδικας ήταν πολύ απλοποιημένος σε πολλές περιπτώσεις, αλλά δεν ήταν λάθος. Το κύριο πρόβλημα ήταν ότι το LLM χρησιμοποίησε συνεχώς assertions στα βήματα GIVEN, ενώ έπρεπε να δημιουργήσει τις απαραίτητες προϋποθέσεις για να τρέξουν τα σενάρια. Με τα assertions, υποθέτει ότι οι προϋποθέσεις έχουν ήδη ολοκληρωθεί κάπου αλλού. Στο τελευταίο βήμα, όπου ζήτησα να δημιουργήσει τη λογική του συστήματος, το LLM δημιούργησε μια κλάση βιβλιοθήκης που αποθηκεύει οντότητες. Αυτό είναι σε αντίθεση με τη χρήση DAOs που είχε υιοθετήσει νωρίτερα. Επίσης, δεν χρησιμοποίησε την κλάση Item και κυρίως χρησιμοποίησε την κλάση Book, επειδή δεν κατάλαβε τη σύνδεση μεταξύ τους (σύνδεσμος πλήρους συνομιλίας).

8.3.2 GPT -3.5 Συνομιλία 2

Περιγραφή:

Σ' αυτή τη συνομιλία, παρείχα στο LLM τις ίδιες ακριβώς πληροφορίες που έδωσα στη φάση 3, Συνομιλία 1, αλλά με μερικές αλλαγές. Αντί για τον όρο “fields,” χρησιμοποίησα τον όρο “attributes,” κάτι που δεν επηρεάζει το αποτέλεσμα. Το πιο σημαντικό είναι ότι, σε αντίθεση με την πρώτη συνομιλία, ζήτησα από το LLM να δημιουργήσει πρώτα τις κλάσεις domain. Είναι αξιοσημείωτο ότι το LLM κατάλαβε εξαιρετικά καλά την εργασία. Παρείχε όλες τις κλάσεις με τις ιδιότητες ακριβώς όπως τις είχα δώσει και δημιούργησε κάποιες "βοηθητικές" κλάσεις όπως “TelephoneNumber,” “Address,” “Money,” “Currency,” και “SystemDate” μόνο από τις ιδιότητες που είχα δώσει. Αν και δεν τις είχα αναφέρει συγκεκριμένα, το LLM κατάλαβε την ανάγκη για αυτές τις κλάσεις και τις δημιούργησε με σωστά αποτελέσματα. Ειδικότερα, δημιούργησε το Enum για το ItemState και εκτίμησε σωστά τη μεταβλητή “currency” που χρειαζόταν στην κλάση “Money,” χωρίς να δοθούν συγκεκριμένες οδηγίες. Ωστόσο, το LLM δεν χρησιμοποίησε καθόλου Services ή DAOs (θα δώσω οδηγίες για αυτά στην επόμενη συνομιλία). Αντί αυτού,

χρησιμοποίησε μια κλάση με το όνομα LibrarySystem για την αποθήκευση, διαγραφή και πρόσβαση σε οντότητες, καθώς και για τη χρήση μεθόδων παρόμοιες με τα Services. Αν και τα αποτελέσματα ήταν σωστά με βάση το σύστημα που το LLM θεώρησε κατάλληλο, έπρεπε να είχε χρησιμοποιήσει DAOs και Services όπως είχα ζητήσει. Επίσης, δεν χρησιμοποίησε καθόλου την κλάση Item, όπως στην φάση 3, Συνομιλία 1, και χρησιμοποίησε μόνο την κλάση Book (σύνδεσμος πλήρους συνομιλίας).

8.3.3 GPT -3.5 Συνομιλία 3

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρείχα στο LLM όλες τις ίδιες πληροφορίες όπως στις άλλες συνομιλίες της φάσης 3. Ωστόσο, ζήτησα να δημιουργήσει και τον κώδικα για τα Services και τα DAOs μαζί με τον κώδικα για τις κλάσεις domain (σε άλλες συνομιλίες, είχα ζητήσει μόνο τον κώδικα για τις κλάσεις domain). Κάτι τέτοιο είχα κάνει επίσης στη φάση 2, Συνομιλία 1 και στη φάση 2, Συνομιλία 4. Τα αποτελέσματα ήταν πολύ καλά για τον κώδικα των κλάσεων domain, καθώς και για τα DAOs. Ωστόσο, για τα Services, το LLM δημιούργησε μερικές παραπάνω κλάσεις από όσες ήταν απαραίτητες, αν και οι κλάσεις αυτές ήταν ακόμα καλές. Στη συνέχεια, το LLM άρχισε αμέσως να δημιουργεί κώδικα (όπως συνήθως κάνει όταν του ζητώ πρώτα να δημιουργήσει τον κώδικα για τις κλάσεις domain), και τα αποτελέσματα ήταν αρκετά καλά σε σύγκριση με τις προηγούμενες συνομιλίες. Ειδικότερα, το LLM κατάλαβε ότι ο "George Red" είναι μια persona και δεν τον προσδιόρισε ως κανονική έκφραση, αλλά δημιούργησε μια μεταβλητή με αυτό το όνομα. Επίσης, χρησιμοποίησε την κλάση Item όπως χρειαζόταν (αυτή τη φορά ξέχασε την κλάση Book, ενώ σε άλλες συνομιλίες είχε ξεχάσει την κλάση Item) και χρησιμοποίησε τις κλάσεις EmailService για τη λειτουργία ειδοποίησης καθυστέρησης. Συνολικά, τα αποτελέσματα αυτής της συνομιλίας ήταν πολύ καλά και χρήσιμα με βάση τον τρέχοντα κώδικα, με κάποιες μικρές αλλαγές. Φαίνεται ότι η υπενθύμιση στο LLM να δημιουργήσει τον κώδικα για τα DAOs και τα Services το βοηθάει να τα χρησιμοποιεί περισσότερο, σε σύγκριση με το να μην αναφέρεται τίποτα (σε άλλες συνομιλίες, απλά ζητούσα από το LLM να τα χρησιμοποιεί στην αρχιτεκτονική του στην πρώτη εντολή) (σύνδεσμος πλήρους συνομιλίας).

8.3.4 GPT -3.5 Συνομιλία 4

Περιγραφή:

Σ' αυτή τη συνομιλία, έδωσα τις ακριβείς ίδιες πληροφορίες όπως και σε όλες τις προηγούμενες, αλλά προσπάθησα να δώσω τα features ένα-ένα. Για κάθε feature, ζήτησα από το LLM να παρέχει τον κώδικα για τις κλάσεις domain καθώς και για τις κλάσεις DAOs/Services. Η προσέγγιση της παροχής των features ένα-ένα φαίνεται να βοηθά το LLM να επικεντρώνεται περισσότερο σε κάθε συγκεκριμένο feature και να δημιουργεί τις κλάσεις Services πολύ καλύτερα από ό,τι στις προηγούμενες συνομιλίες. Οι κλάσεις domain και DAO είναι ακριβώς οι ίδιες, κάτι που είναι θετικό. Ωστόσο, το LLM δεν δημιούργησε τόσο πολύ κώδικα όσο στις άλλες συνομιλίες, κυρίως επειδή περιμένει τα υπόλοιπα features για να του δώσουν ενδείξεις. Έτσι, αναφέρει στα σχόλια ότι πρέπει να υλοποιηθεί η λογική για το feature, ενώ στις άλλες συνομιλίες παρείχε κώδικα για όλα. Συνολικά, ο κώδικας ήταν σε πολλές περιπτώσεις ο ίδιος με τον κώδικα των άλλων συνομιλιών και μερικές φορές καλύτερος, αλλά δεν είμαι σίγουρος αν αυτή η μέθοδος οδηγεί πάντα σε καλύτερο κώδικα. Παρόλο που υπάρχουν πλεονεκτήματα, όπως η καλύτερη εστίαση σε κάθε feature, υπάρχουν και μειονεκτήματα, όπως η έλλειψη ολοκληρωμένης υλοποίησης για κάθε feature. Επιπλέον, η μέθοδος αυτή μπορεί να μην είναι πάντα η καλύτερη, καθώς ένα σενάριο που περιγράφεται στο τελευταίο feature μπορεί να επηρεάσει τη σωστή υλοποίηση ενός σεναρίου στο πρώτο feature (σύνδεσμος πλήρους συνομιλίας).

8.3.5 GitHub Copilot Συνομιλία 1

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρέδωσα στο LLM τα features του συστήματος και τις κλάσεις domain με τις ιδιότητές τους.. Το Copilot κατάλαβε πολύ καλά την εργασία και άρχισε να δημιουργεί κώδικα αμέσως, χωρίς να χρειαστεί να του ζητήσω ρητά να δημιουργήσει κώδικα για τις κλάσεις domain. Στη συνέχεια, με μερικές επιπλέον μικρές εντολές, δημιούργησε πολύ γρήγορα όλα τα step definitions. Συνολικά, το LLM χρησιμοποίησε πολλές από τις ιδιότητες των κλάσεων που του παρείχα και κατάλαβε πολύ καλύτερα πώς να τα χρησιμοποιήσει σε σύγκριση με το GPT-3.5. Χρησιμοποίησε επίσης τα Services και τα DAOs, αν και τα αποτελέσματα δεν ήταν εξαιρετικά καλύτερα από τις συνομιλίες της φάσης 2, κυρίως λόγω της χρήσης πολλών assertions στα βήματα GIVEN. Στην επόμενη συνομιλία, θα ζητήσω από το LLM να μην χρησιμοποιεί assertions στα βήματα GIVEN και θα δούμε τα αποτελέσματα. Παρά ταύτα, πολλά από τα step definitions ήταν πολύ χρηστικά (σύνδεσμος πλήρους συνομιλίας).

8.3.6 GitHub Copilot Συνομιλία 2

Περιγραφή:

Σ' αυτήν τη συνομιλία της φάσης 3, παρέδωσα τις ακριβώς ίδιες πληροφορίες και αρχικές εντολές όπως στη Συνομιλία 1, αλλά ζήτησα από το LLM να μην χρησιμοποιεί assertions στα βήματα GIVEN και εξήγησα τον λόγο. Το Copilot το κατάλαβε πολύ καλά, δημιούργησε μόνο του τις κλάσεις domain (που φαίνεται να βοηθά στη μνήμη του αργότερα) και παρήγαγε πολύ καλά αποτελέσματα για τα step definitions, ίσως από τα καλύτερα που έχουμε δει. Το LLM έκανε εξαιρετική χρήση των ιδιοτήτων που του παρείχα, ιδιαίτερα με την κλάση item / book. Σε προηγούμενες συνομιλίες, δεν είχε χρησιμοποιήσει καθόλου την κλάση book, αλλά τώρα θυμήθηκε τη σύνδεσή της με την κλάση item και δημιούργησε πολύ καλές συνδέσεις. Επιπλέον, κατανόησε πολύ καλά την persona George Red και τη δημιουργία των dummy items, όπως είχε κάνει σε πολλές προηγούμενες φάσεις. Συνολικά, ο παραγόμενος κώδικας ήταν εξαιρετικός: απλός αλλά αποτελεσματικός σε πολλές περιπτώσεις και αρκετά κοντά στην πραγματικότητα. Το LLM χρησιμοποίησε κάποια διαφορετικά Services, αλλά αυτό ήταν αποδεκτό και δεν επηρεάζει αρνητικά την ποιότητα του παραγόμενου κώδικα (σύνδεσμος πλήρους συνομιλίας).

8.3.7 GitHub Copilot Συνομιλία 3

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρείχα στο Copilot τις ακριβώς ίδιες πληροφορίες όπως στις συνομιλίες 2 και 3, αλλά ζήτησα να δημιουργήσει πρώτα τον κώδικα για τις κλάσεις domain (ενώ σε άλλες συνομιλίες δεν είχα δώσει μια τέτοια εντολή). Το LLM κατάλαβε αρκετά καλά πώς και ποιες κλάσεις domain να δημιουργήσει. Ωστόσο, δεν χρησιμοποίησε DAOs ή Services και χρησιμοποίησε μια κλάση με το όνομα LibrarySystem για όλα αυτά, η οποία δεν χρησιμοποιούσε DAOs. Συνολικά, τα αποτελέσματα ήταν καλά, αλλά η χρήση αυτής της κλάσης LibrarySystem δεν ήταν ιδανική. Στην επόμενη συνομιλία, θα ζητήσω από το LLM να δημιουργήσει τον κώδικα για τα DAOs και τα Services από την αρχή, για να δω αν αυτό θα βοηθήσει το σύστημα να τα χρησιμοποιήσει περισσότερο, όπως είχε γίνει με το GPT-3.5, το οποίο είχε θετικά αποτελέσματα (σύνδεσμος πλήρους συνομιλίας).

8.3.8 GitHub Copilot Συνομιλία 4

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρέδωσα στο LLM τις ίδιες πληροφορίες όπως και σε όλες τις προηγούμενες συνομιλίες, αλλά ζήτησα να μου δώσει τον κώδικα για τα DAOs και τα Services μαζί με τον κώδικα για τις κλάσεις domain. Όπως αναμενόταν, το LLM δημιούργησε αρκετά καλό κώδικα για τις κλάσεις domain, καθώς και για τα Services και τα DAOs. Ωστόσο, ο κώδικας που παρήγαγε δεν ήταν πολύ καλός και υπήρχαν θέματα, με αποτέλεσμα τη δημιουργία λιγότερου κώδικα από όσο ήταν επιθυμητό. Συνολικά, η συνομιλία αυτή χρησιμοποιήθηκε για να εξετάσουμε αν το LLM θα χρησιμοποιήσει περισσότερα Services και DAOs, κάτι που το έκανε, αλλά δεν εστιάσαμε τόσο στη ποιότητα του κώδικα καθ' εαυτή (σύνδεσμος πλήρους συνομιλίας).

8.3.9 GPT -4 Συνομιλία 1

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM όλα τα features του συστήματος, τα ονόματα των κλάσεων domain που θα χρειαστεί να χρησιμοποιήσει, καθώς και την αρχιτεκτονική και γενικές γνώσεις του συστήματος. Το LLM κατάλαβε αμέσως την εργασία και δημιούργησε τα features με ελάχιστη καθοδήγηση. Ωστόσο, αρχικά ξέχασε να χρησιμοποιήσει τα DAOs, παρά τις ρητές οδηγίες που του είχα δώσει, και άρχισε να τα χρησιμοποιεί μόνο αφού το ανέφερα ξανά. Γενικά, το LLM χρησιμοποίησε σωστά τις ιδιότητες που παρείχα, αλλά παρουσίασε μερικές μη ρεαλιστικές ή υπερβολικά περίπλοκες απαντήσεις για κάποιες από τις βήμα προς βήμα ορισμούς. Αυτό μπορεί να επηρεάσει την αναγνωσιμότητα και την ευκολία τροποποίησης των βημάτων (σύνδεσμος πλήρους συνομιλίας).

8.3.10 GPT -4 Συνομιλία 2

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM όλα τα features του συστήματος, τα ονόματα των κλάσεων Domain καθώς και τις ιδιότητές τους, και την αρχιτεκτονική/γενικές γνώσεις του συστήματος. Επίσης, έδωσα οδηγίες στο LLM να δημιουργήσει πρώτα τον κώδικα για τις κλάσεις Domain, DAOs και Services. Το LLM κατανόησε αρκετά καλά την εργασία και δημιούργησε τον απαιτούμενο κώδικα με ελάχιστη καθοδήγηση. Παρήγαγε όλα τα features του συστήματος με λεπτομέρεια, και οι απαντήσεις ήταν εύκολες στην κατανόηση. Χρησιμοποίησε σωστά τα χαρακτηριστικά που παρείχα και, συνολικά, ακολούθησε την αρχιτεκτονική που του είχα υποδείξει (σύνδεσμος πλήρους συνομιλίας).

8.3.11 GPT -4 Συνομιλία 3

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM τα features του συστήματος, τα ονόματα των κλάσεων τομέα και τις ιδιότητές τους, καθώς και την αρχιτεκτονική και γενικές γνώσεις του συστήματος. Το LLM κατανόησε αρκετά καλά την εργασία, δημιούργησε κώδικα με πολύ λίγη καθοδήγηση και παρείχε απλές αλλά καλές απαντήσεις. Ωστόσο, το πρόβλημα, όπως συνέβη και σε άλλες συνομιλίες που χρησιμοποίησα αυτή την τεχνική, είναι ότι το LLM θα ωφελούνταν περισσότερο αν είχε όλες τις πληροφορίες ταυτόχρονα αντί να τις λαμβάνει σταδιακά. Πολλά features αλληλοσυνδέονται και προσφέρουν πολύτιμες πληροφορίες που μπορεί να βελτιώσουν την ακρίβεια και τη συνοχή του παραγόμενου κώδικα (σύνδεσμος πλήρους συνομιλίας).

8.3.12 GPT -4o Συνομιλία 1

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM τα features του συστήματος, τις κλάσεις Domain του συστήματος με τις ιδιότητές τους, και την περιγραφή του συστήματος. Όπως σε κάθε άλλη φάση με αυτό το LLM, το LLM κατάλαβε εξαιρετικά γρήγορα τι έπρεπε να κάνει και πώς να συνδέσει τις κλάσεις. Δημιούργησε όλο τον κώδικα για τις κλάσεις Domain, Services και DAOs χωρίς να χρειαστεί καθοδήγηση, και στη συνέχεια, με μόνο 2 μηνύματα, δημιούργησε όλα τα Step Definitions για όλα τα features. Συνολικά, ο κώδικας ήταν αρκετά καλός, πολλές φορές πολύ κοντά στον πραγματικό κώδικα του συστήματος, και περιλάμβανε πολλές λεπτομέρειες σε κάθε βήμα, χωρίς να καταφύγει στην εύκολη λύση με απλές δηλώσεις ή εκτυπώσεις στους ορισμούς βημάτων (σύνδεσμος πλήρους συνομιλίας).

8.3.13 GPT -4o Συνομιλία 2

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM τις ακριβώς ίδιες πληροφορίες όπως στη Συνομιλία 1, με τη μικρή διαφορά ότι το καθοδήγησα να δημιουργήσει πρώτα τον κώδικα για τα Domain, Services και DAOs, και στη συνέχεια να δημιουργήσει τον κώδικα για τα Step Definitions. Για άλλη μια φορά, το LLM κατάλαβε καλά την εργασία, δημιούργησε καλό κώδικα για τα Domain, Services και DAOs με πολλές λεπτομέρειες, και με πολύ λίγα μηνύματα (2-3 μηνύματα) δημιούργησε όλα τα Step Definitions χωρίς να παραλείψει κανένα. Αυτό είναι πολύ σημαντικό για

τον χρήστη από άποψη χρόνου. Τα αποτελέσματα ήταν αρκετά καλά με αυτές τις συγκεκριμένες γνώσεις και πολύ χρήσιμα σε πολλές περιπτώσεις (σύνδεσμος πλήρους συνομιλίας).

8.3.14 GPT -4o Συνομιλία 3

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM ακριβώς τις ίδιες πληροφορίες όπως στις συνομιλίες 1 και 2 αυτής της φάσης, αλλά έδωσα όλες τις πληροφορίες σε μόνο 2 μηνύματα αντί να τις χωρίσω. Το LLM κατάφερε να διαχειριστεί όλες τις πληροφορίες με ευκολία. Τα αποτελέσματα ήταν παρόμοια με αυτά των συνομιλιών 1 και 2, με μικρές διαφορές σε κάποιες περιοχές (σύνδεσμος πλήρους συνομιλίας).

8.4 Φάση 4

8.4.1 GPT -3.5 Συνομιλία 1

Περιγραφή:

Σε αυτή τη συνομιλία της φάσης 4, παρείχα στο LLM όλες τις κλάσεις domain, τις ιδιότητες κάθε κλάσης, τα ονόματα των συναρτήσεων και τα features του συστήματος. Αρχικά, ζήτησα από το LLM να παρέχει τον κώδικα για τις κλάσεις domain, DAOs και services. Το LLM αντέδρασε πολύ καλά στην εντολή μου, παρέχοντας κάθε κλάση και τα χαρακτηριστικά/συναρτήσεις όπως τα είχα δώσει. Έκανε αρκετά καλή δουλειά με τα DAOs και κάπως καλή με τα services. Ωστόσο, συνολικά, το LLM χρησιμοποίησε ελάχιστα τις συναρτήσεις που είχα παρέχει και, όπως και σε προηγούμενες συνομιλίες, ακολούθησε τον δικό του τρόπο συγγραφής του κώδικα. Αν και δεν ήταν λάθος, ήταν αρκετά απλοποιημένος σε πολλές περιπτώσεις, κάτι που σε πραγματικό σύστημα θα μπορούσε να οδηγήσει σε λάθη. Επομένως, φαίνεται ότι το LLM μπορεί να μην είναι σε θέση να θυμάται πολλές πληροφορίες για όλα τα χαρακτηριστικά και τις συναρτήσεις. Ενώ οι ιδιότητες σίγουρα βοήθησαν το LLM, οι συναρτήσεις δεν φαίνεται να επηρεάσαν σχεδόν καθόλου την ποιότητα του παραγόμενου κώδικα (σύνδεσμος πλήρους συνομιλίας).

8.4.2 GPT -3.5 Συνομιλία 2

Περιγραφή:

Σε αυτή τη συνομιλία, παρείχα στο LLM ακριβώς τις ίδιες πληροφορίες όπως στη Φάση 4, Συνομιλία 1 (features, κλάσεις domain με ιδιότητες και μεθόδους, και την αρχιτεκτονική), αλλά

άλλαξα τη λέξη "functions" σε "methods". Αρχικά, ζήτησα από το LLM να παρέχει τον κώδικα για τις κλάσεις domain, DAOs και services, κάτι που το LLM έκανε πολύ καλά, όπως και σε όλες τις άλλες συνομιλίες. Στη συνέχεια, ζήτησα συγκεκριμένα από το LLM να θυμάται και να χρησιμοποιεί τις μεθόδους/ιδιότητες που του παρείχα για κάθε κλάση. Τα αποτελέσματα ήταν κατά πολύ καλύτερα αυτή τη φορά. Το LLM πραγματικά κατάλαβε την εργασία, δεν παρέλειψε κανένα από τα step definitions (αν και σε μερικά έδωσε πολύ απλές απαντήσεις) και χρησιμοποίησε πολλές από τις μεθόδους που του είχα παρέχει, όπως για παράδειγμα για την καταμέτρηση των εκκρεμών αντικειμένων. Επίσης, σχεδόν μάντεψε σωστά και παρείχε χρήσιμα/εκτελέσιμα αποτελέσματα για πολλές από τις step definitions, όπως το "@Given("^George Red has (\\d+) pending items to be returned")". Συνολικά, τα αποτελέσματα αυτής της συνομιλίας ήταν εξαιρετικά (σύνδεσμος πλήρους συνομιλίας).

8.4.3 GPT -3.5 Συνομιλία 3

Περιγραφή:

Αυτή η συνομιλία δεν ήταν τόσο σημαντική. Παρείχα ακριβώς τις ίδιες πληροφορίες σε κάθε σημείο, όπως στη Φάση 4, Συνομιλία 2, μόνο για να ελέγξω αν με τις ίδιες ή παρόμοιες εντολές το LLM θα δημιουργήσει αποτελέσματα εξίσου καλά με αυτά της Συνομιλίας 2. Σε πολλές περιπτώσεις, το LLM παρείχε αποτελέσματα που ήταν αρκετά καλά, αν και όχι ακριβώς τόσο καλά όσο στη Συνομιλία 2. Αυτό δείχνει ότι το επίπεδο των αποτελεσμάτων μπορεί να έχει επηρεαστεί από τυχαίους παράγοντες στην προηγούμενη συνομιλία (σύνδεσμος πλήρους συνομιλίας).

8.4.4 GPT -3.5 Συνομιλία 4

Περιγραφή:

Σε αυτή τη συνομιλία, παρείχα στο LLM τις ίδιες πληροφορίες όπως σε όλες τις άλλες συνομιλίες, αλλά έδωσα τα features ένα προς ένα (όπως κάνω σε κάθε φάση). Ο κώδικας για τις κλάσεις domain, κ.λπ. ήταν σχεδόν ίδιος με αυτόν σε όλες τις άλλες συνομιλίες, αλλά ο κώδικας για τα step definitions δεν ήταν τόσο καλός σε σύγκριση με προηγούμενες συνομιλίες (χρησιμοποίησε μια κλάση βιβλιοθήκης αντί για τα DAOs). Το συμπέρασμά μου είναι ότι, καθώς υπάρχουν μερικές περισσότερες εντολές με τις μεθόδους, το LLM φαίνεται να ξεχνά τα DAOs όταν αυτά έχουν αναφερθεί πολλές μηνύματα πριν. Όταν δίνω τα τελευταία features, φαίνεται να τα έχει ξεχάσει

και χρησιμοποιεί διαφορετικές κλάσεις. Αυτό δεν συμβαίνει αν δώσω τα features όλα μαζί από την αρχή, καθώς το LLM παίρνει τις πληροφορίες από την αρχή και στη συνέχεια υλοποιεί μόνο τον κώδικα, χωρίς να χρειάζεται να λάβει επιπλέον πληροφορίες. Επιπλέον, δεν δημιουργήθηκαν κάποια DAOs γιατί δεν είχε τα χαρακτηριστικά που παρέχουν τη λύση για ορισμένες υλοποιήσεις DAO (σύνδεσμος πλήρους συνομιλίας).

8.4.5 GitHub Copilot Συνομιλία 1

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρέδωσα στο Copilot όλα τα features του συστήματος, καθώς και τον κώδικα για τις κλάσεις domain με τις ιδιότητες και τις μεθόδους τους. Ζήτησα από το LLM να μου παρέχει πρώτα τον κώδικα για τις κλάσεις domain, καθώς και τον κώδικα για τα DAOs και τα Services, με σκοπό να χρησιμοποιήσει περισσότερο τα DAOs. Το LLM έκανε αρκετά καλά αυτή τη δουλειά (πολύ παρόμοιο με τις απαντήσεις του GPT για τις μεθόδους του `friendLoanDomain`). Ωστόσο, τα αποτελέσματα δεν ήταν ιδιαίτερα εντυπωσιακά και ήταν πολύ παρόμοια με εκείνα της φάσης 3. Αυτό σημαίνει ότι το LLM δεν χρησιμοποίησε πραγματικά τις μεθόδους ή τις χρησιμοποίησε σε περιορισμένες περιπτώσεις (αν και χρησιμοποίησε μερικές). Επιπλέον, το LLM δεν παρείχε κάποια Step Definitions μέχρι να το ζητήσω πολλαπλές φορές και χρησιμοποίησε ξανά assertions στα βήματα GIVEN, παρά το γεγονός ότι είχα ζητήσει να μην το κάνει. Συνολικά, τα αποτελέσματα ήταν μέτρια με βάση τις πληροφορίες που δόθηκαν (σύνδεσμος πλήρους συνομιλίας).

8.4.6 GitHub Copilot Συνομιλία 2

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρέδωσα στην Copilot τις ακριβείς πληροφορίες όπως στη Συνομιλία 1, αλλά ζήτησα να χρησιμοποιήσει όσο το δυνατόν περισσότερες από τις μεθόδους και τις ιδιότητες για τη δημιουργία των step definitions. Αυτή η εντολή είχε βοηθήσει πολύ το GPT-3.5, οπότε σκέφτηκα ότι θα ήταν καλή ιδέα να τη δοκιμάσω και με αυτό το LLM. Συνολικά, ο κώδικας για τις κλάσεις domain ήταν καλός, όπως πάντα. Ωστόσο, τα step definitions δεν ήταν κάτι ιδιαίτερο και τα αποτελέσματα ήταν μέτρια. Το LLM χρησιμοποίησε ξανά assertions στα βήματα GIVEN, κάτι που δεν είναι ιδανικό και απλοποιεί τα πράγματα περισσότερο από όσο θα έπρεπε. Παρά το

γεγονός ότι το LLM χρησιμοποίησε πολλές από τις μεθόδους που της παρέδωσα, τα συνολικά αποτελέσματα παρέμειναν μέτρια (σύνδεσμος πλήρους συνομιλίας).

8.4.7 GitHub Copilot Συνομιλία 3

Περιγραφή:

Σ' αυτήν τη συνομιλία, παρέδωσα στο LLM τις ακριβείς πληροφορίες όπως και στις προηγούμενες συνομιλίες, αλλά δεν του έδωσα την εντολή να δημιουργήσει πρώτα τον κώδικα για τις κλάσεις domain/DAOs/Services, για να διαπιστώσω αν αυτό θα είχε κάποια επίδραση στην παραγωγή του κώδικα. Αυτή η αλλαγή δεν είχε καμία σημαντική διαφορά, καθώς το LLM δημιούργησε σχεδόν τον ίδιο κώδικα για το πρώτο feature. Έτσι, συνέχισα την συνομιλία μέχρι να παραχθεί όλος ο κώδικας (σύνδεσμος πλήρους συνομιλίας).

8.4.8 GPT -4 Συνομιλία 1

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM όλα τα features του συστήματος, τα ονόματα, τις ιδιότητες και τις μεθόδους όλων των κλάσεων τομέα, καθώς και την αρχιτεκτονική του συστήματος. Ζήτησα από το LLM να δημιουργήσει πρώτα τον κώδικα για τις κλάσεις Domain, DAOs και Services, και το έκανε εξαιρετικά. Το LLM κατανόησε πολύ καλά το σύστημα, δημιούργησε με ευχέρεια τα περισσότερα Step Definitions και παρείχε καλό κώδικα για πολλά από αυτά. Συνολικά, όλες αυτές οι πληροφορίες βοήθησαν το LLM να παράγει χρήσιμα αποτελέσματα σε πολλά από τα Step Definitions, δείχνοντας την αποτελεσματικότητα της διαδικασίας παροχής ολοκληρωμένων πληροφοριών από την αρχή (σύνδεσμος πλήρους συνομιλίας).

8.4.9 GPT -4 Συνομιλία 2

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM όλα τα features του συστήματος, τα ονόματα, τις ιδιότητες και τις μεθόδους των κλάσεων Domain, καθώς και την αρχιτεκτονική και γενικές γνώσεις του συστήματος. Το LLM δημιούργησε αρκετά καλά αποτελέσματα, αν και χρειάστηκε κάποιες επαναλαμβανόμενες καθοδηγήσεις για να θυμηθεί να χρησιμοποιήσει τα DAOs και να δημιουργήσει όλα τα Step Definitions. Παρά αυτές τις μικρές δυσκολίες, τα αποτελέσματα ήταν

παρόμοια με εκείνα της προηγούμενης συνομιλίας αυτής της φάσης και ήταν ικανοποιητικά (σύνδεσμος πλήρους συνομιλίας).

8.4.10 GPT -4 Συνομιλία 3

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM όλα τα features του συστήματος ένα προς ένα, μαζί με τα ονόματα, τις ιδιότητες και τις μεθόδους των κλάσεων τομέα, καθώς και την αρχιτεκτονική και τις γενικές γνώσεις του συστήματος. Το LLM κατανόησε εύκολα την εργασία και χρειάστηκε πολύ λίγα μηνύματα για την παραγωγή των αποτελεσμάτων. Παρά την καλή κατανόηση, τα αποτελέσματα ήταν πιο αδύναμα σε σύγκριση με άλλες συνομιλίες (σύνδεσμος πλήρους συνομιλίας).

8.4.11 GPT -4o Συνομιλία 1

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM όλα τα features του συστήματος, τις κλάσεις Domain με τις αντίστοιχες ιδιότητες και τα ονόματα μεθόδων, καθώς και την περιγραφή του συστήματος. Ζήτησα από το LLM να παρέχει πρώτα τον κώδικα για το Domain, κάτι που δεν είναι απαραίτητο, καθώς το LLM το κάνει αυτό αυτόματα, ακόμα κι αν δεν το καθοδηγήσω. Όπως πάντα, το LLM κατάλαβε εξαιρετικά καλά την εργασία, παρείχε τον κώδικα για το Domain, DAOs και Services με πολλές λεπτομέρειες, και στη συνέχεια μου έδωσε τον κώδικα για τα Step Definitions. Σε πολλές περιπτώσεις, ο κώδικας ήταν ακριβώς ίδιος με την πραγματική λύση, ενώ σε κάποιες άλλες περιπτώσεις παρήγαγε ακόμα καλύτερες λύσεις. Συνολικά, τα αποτελέσματα ήταν αρκετά καλά, χρειάστηκε πολύ λίγα μηνύματα, και το LLM είχε πολύ καλή κατανόηση του τι έπρεπε να κάνει (σύνδεσμος πλήρους συνομιλίας).

8.4.12 GPT -4o Συνομιλία 2

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM τις ίδιες ακριβώς πληροφορίες όπως στη Συνομιλία 1, αλλά δεν το καθοδήγησα να δημιουργήσει πρώτα τον κώδικα για το Domain, Services και DAOs. Ωστόσο, όπως σε όλες τις προηγούμενες συνομιλίες, το LLM τα δημιούργησε από μόνο του. Τα αποτελέσματα του κώδικα ήταν αρκετά καλά και παρόμοια με αυτά των άλλων συνομιλιών. Το LLM χρησιμοποίησε την αρχιτεκτονική που του είχα δώσει στην αρχή της συνομιλίας, παρείχε

πολλές λεπτομέρειες στις απαντήσεις του και χρειάστηκε πολύ λίγα μηνύματα για να δημιουργήσει όλα τα Step Definitions για όλα τα features που παρείχα (σύνδεσμος πλήρους συνομιλίας).

8.4.13 GPT -4 Συνομιλία 3

Περιγραφή:

Στη συνομιλία αυτή, παρέθεσα στο LLM τις ίδιες ακριβώς πληροφορίες όπως στη Συνομιλία 2, αλλά αποφάσισα να δώσω όλες τις κλάσεις Domain, Services, DAOs, τις ιδιότητες, τις μεθόδους και τα features σε ένα μόνο μήνυμα, για να δω αν το LLM μπορεί να κατανοήσει όλη τη γνώση χωρίς να είναι χωρισμένη σε πολλά μηνύματα. Συνολικά, το LLM αντέδρασε ακριβώς όπως και σε άλλες συνομιλίες. Τα αποτελέσματα ήταν αρκετά καλά, αλλά σε μερικές περιπτώσεις ήταν λιγότερο αποτελεσματικά σε σύγκριση με τις Συνομιλίες 1 και 2. Ωστόσο, χρειάστηκε πολύ λίγα μηνύματα για να δημιουργήσει όλα τα Step Definitions για όλα τα features (σύνδεσμος πλήρους συνομιλίας).