ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εξαγωγή και Ανάλυση Δεδομένων Επαναχρησιμοποίησης Λογισμικού από το Αποθετήριο GitHub

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΡΗΣΤΟΥ ΟΡΦΕΑ

Επιβλέπουσα: Μπίμπη Σταματία

Επίκουρη Καθηγήτρια

ΚΟΖΑΝΗ/ΦΕΒΡΟΥΑΡΙΟΣ/2023

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL
& COMPUTER ENGINEERING
ece.uowm.gr

Extraction and Analysis of Reuse Software Data From Repository of GitHub

THESIS

CHRISTOU ORFEAS

SUPERVISOR: Bibi Stamatia

Assistant Professor

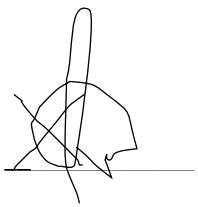
KOZANI/FEBRUARY/2023

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Δήλωση μη Λογοκλοπής και Ανάληψης Προσωπικής Ευθύνης

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο " **Εξαγωγή και Ανάλυση Δεδομένων Επαναχρησιμοποίησης Λογισμικού από το Αποθετήριο GitHub**" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κα. **Μπίμπη Σταματία** αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή: 

Περίληψη

Η JavaScript αποτελεί την βασική γλώσσα σεναρίων στην ανάπτυξη ιστοσελίδων στην σημερινή εποχή και η διάδοση της αυξάνεται με σταθερό ρυθμό. Ωστόσο η συνεχής ανάπτυξη συνδέεται άμεσα με την αύξηση της πολυπλοκότητας. Έτσι το μεγαλύτερο ποσοστό των προγραμματιστών επιλέγει την μέθοδο της επαναχρησιμοποίησης στοιχείων είτε με την χρήση βιβλιοθηκών είτε πλαισίων με σκοπό να επιτύχει την αύξηση της αποδοτικότητας και την εξοικονόμηση χρόνου. Η μελέτη αυτή με την βοήθεια κατάλληλων μετρικών εξετάζει εάν υπάρχει πρακτική εφαρμογή των νόμων του Lehman στα έργα της σημερινής εποχής. Στο πλαίσιο αυτό μελετήθηκαν 80 δημοφιλή έργα JavaScript από το αποθετήριο GitHub και περισσότερες από 2,000 εκδόσεις .Συνεπώς τα αποτελέσματα έδειξαν ότι μόνο οι νόμοι της Διατήρησης Οικειότητας και Συνεχούς Ανάπτυξης επιβεβαιώθηκαν και οι υπόλοιποι από τους 7 που μελετήθηκαν απορρίφθηκαν. Κατά γενική ομολογία καταλήξαμε ότι οι προγραμματιστές αποφεύγουν τις πολλές και μεγάλες αλλαγές στα συστήματα τους και αντίθετα επιλέγουν εκδόσεις τις οποίες εμπιστεύονται και έχουν οικειότητα.

Λέξεις Κλειδιά

JavaScript, Επαναχρησιμοποίηση, Νόμοι Lehman, GitHub

Abstract

JavaScript is the most common scripting language for developing a website and her popularity increases more and more in these days. Although, this rise has an effect in increasing the complexity of the websites. Therefore, most developers lean towards reusable components such as libraries and frameworks to boost their productivity and preserve time. In this paper we use metrics to investigate if Lehman's Laws have an actual effect in today's projects. Hence, we studied 80 popular JavaScript projects from GitHub repository and more than 2,000 versions. We concluded that we can only verify Conservation of Familiarity and Continuing Growth and reject the other 7 laws. In general speaking developers tend to avoid making huge changes during the updates, instead they are more likely to use versions which are more familiar with.

Keywords

JavaScript, Reusable Components, Lehman's Laws, GitHub

Ευχαριστίες

Ευχαριστώ τους γονείς μου και τα αδέρφια μου που με στήριξαν καθ’ όλη την διάρκεια της φοιτητικής μου πορείας.

Επίσης θα ήθελα να ευχαριστήσω τους φίλους μου που χωρίς αυτού δεν θα έφτανα ως εδώ

Τέλος ευχαριστώ την επιβλέπουσα καθηγήτρια κα Μπίμπη και την κα Τερζή που με εμπιστεύθηκαν να φέρω εις πέρας αυτό το εγχείρημα.

ΑΥΤΗ Η ΣΕΛΙΔΑ ΕΙΝΑΙ ΣΚΟΠΙΜΑ ΛΕΥΚΗ

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Πρόλογος

Η παρούσα διπλωματική εργασία με τίτλο «**Εξαγωγή και Ανάλυση Δεδομένων Επαναχρησιμοποίησης Λογισμικού από το Αποθετήριο GitHub»** δημιουργήθηκε από τις αρχές Οκτωβρίου 2021 ως τον Δεκέμβριο 2022 στην πόλη της Καστοριάς υπό την επίβλεψη της επίκουρης καθηγήτριας Μπίμπης Σταματίας και της υποψήφιας διδάκτορος Τερζή Αναστασίας. Το παρακάτω πόνημα αποτελεί επέκταση της διπλωματικής εργασίας με τίτλο «**Δυνατότητες Επαναχρησιμοποίησης Λογισμικού σε Έργα JavaScript»** συγγραφέας της οποίας είναι η Τερζή Αναστασία. Η ταχεία διάδοση της γλώσσας JavaScript στην ανάπτυξη ιστοσελίδων και της μεθόδου της επαναχρησιμοποίησης υπάρχουσών βιβλιοθηκών και το πώς αυτές επηρεάζουν την εξέλιξη ενός συστήματος αποτέλεσαν τον κυριότερο παράγοντα για να γεννηθεί η ιδέα να συντάξουμε αυτό το έργο. Αρχικά, για λόγους πρακτικούς το έργο αυτό χωρίστηκε σε δύο στάδια, το ερευνητικό κομμάτι και το συγγραφικό. Το ερευνητικό κομμάτι ήταν το πιο επίπονο και χρονοβόρο από τα δύο καθώς αντιμετωπίσαμε πολλά προβλήματα στην συγγραφή και την εφαρμογή του κώδικα που χρησιμοποιήσαμε για την συλλογή των δεδομένων. Επίσης η συλλογή των δεδομένων ήταν μια χρονοβόρα και απαιτητική διαδικασία κατά την οποία προέκυψαν προβλήματα τα οποία δεν είχαμε υπολογίσει και αναγκαστήκαμε να επιλύσουμε. Αφού η συλλογή ολοκληρώθηκε το επόμενο στάδιο ήταν το φιλτράρισμα των δεδομένων και η στατιστική ανάλυση με την βοήθεια του JASP. Έτσι ολοκληρώθηκε το πρώτο στάδιο της μελέτης που περιείχε κυρίως το πρακτικό κομμάτι της έρευνας. Αντίθετα το συγγραφικό κομμάτι κύλησε ομαλά και χωρίς πολλές δυσκολίες. Αρχικά μελετήσαμε την απαραίτητη βιβλιογραφία για να αποδώσουμε το θεωρητικό υπόβαθρο και να επεξηγήσουμε τις έννοιες που θα συναντήσει ο αναγνώστης στο παρακάτω εγχείρημα. Στην συνέχεια με χρήση εικόνων, πινάκων και σχημάτων παρουσιάσαμε τα δεδομένα που συλλέξαμε στο προηγούμενο στάδιο. Ακόμη παρουσιάσαμε τα συμπεράσματα που προέκυψαν και τα συγκρίναμε με άλλες μελέτες παρόμοιου αντικειμένου και παροτρύναμε τους αναγνώστες για μελλοντικές επεκτάσεις. Τέλος θεωρήσαμε απαραίτητο να αναλύσουμε την εγκυρότητα και την αξιοπιστία της μελέτης αυτής αλλά και να τονίσουμε την ερευνητική αλλά και πρακτική χρησιμότητα αυτού του έργου.

.

Κεφάλαιο 1: Εισαγωγή

Σήμερα σχεδόν όλες οι συσκευές (Windows,Android,iOS,κ.λπ.) χρησιμοποιούν JavaScript.Το έτος 2022 μεταξύ 1,8 δισεκατομμυρίων ιστοσελίδων το 98% αυτών βασίζονται στην JavaScript για τα client-side προγράμματα, γεγονός που την καθιστά μια από τις δημοφιλέστερες γλώσσες προγραμματισμού. Η ελαφριά φύση της σε συνδυασμό με την δυνατότητα να χρησιμοποιηθεί και στην front-end και στην back-end ανάπτυξη μιας ιστοσελίδας είναι οι λόγοι που την καθιστούν τόσο δημοφιλή στην προγραμματιστική κοινότητα. Ακόμη, η JavaScript έχει έναν τεράστιο όγκο βιβλιοθηκών και πλαισίων που βοηθούν τους προγραμματιστές να δημιουργήσουν πολύπλοκες εφαρμογές με λιγότερο κόστος και με μεγαλύτερη ταχύτητα. Με την βοήθεια του GitHub μια από τις δημοφιλέστερες πλατφόρμες για αποθήκευση και κοινοποίηση προγραμματιστικών δεδομένων αλλά και τα αμέτρητα έργα ανοιχτού κώδικα που διαθέτει καθιστούν την επαναχρησιμοποίηση λογισμικού μια από τις βασικότερες μεθόδους ανάπτυξης λογισμικού στην σημερινή εποχή. Ωστόσο η εξέλιξη των έργων για να ανταπεξέλθουν στις νέες απαιτήσεις έχει ως αποτέλεσμα την αύξηση της πολυπλοκότητας και τις αλλαγές στην δομή τους. Οι μεγάλες αλλαγές στην αρχιτεκτονική δομή ενός συστήματος έχει επιρροές και στα έργα τα οποία επαναχρησιμοποιούν στοιχεία αυτού του συστήματος[1]. Τέλος η σωστή χρήση των επαναχρησιμοποιούμενων στοιχείων αποτελεί ακόμη και σήμερα μία πρόκληση για τους προγραμματιστές.

1.1 Αντικείμενο της Διπλωματικής

Σκοπός αυτής της έρευνας είναι να προσπαθήσουμε να κατανοήσουμε πώς η εξέλιξη ενός έργου JavaScript και οι αλλαγές που συμβαίνουν στις επαναχρησιμοποιούμενες βιβλιοθήκες του επηρεάζουν θετικά ή αρνητικά την αξιοπιστία και την αποδοτικότητα αυτού. Αναπτύξαμε μετρικές οι οποίες θα μας βοηθήσουν να εξερευνήσουμε τους Νόμους του Lehman πάνω στην εξέλιξη λογισμικού με απώτερο σκοπό να βοηθήσουμε τους προγραμματιστές να βελτιώσουν το τρόπο και την συχνότητα που αναβαθμίζουν τα έργα τους, ώστε να δημιουργήσουν ένα πλάνο εξέλιξης του έργου που θα αποφέρει έναν αξιόλογο συνδυασμό αξιοπιστίας και απόδοσης με σκοπό την αύξηση των κερδών. Για να καταλήξουμε σε αυτά τα συμπεράσματα χρησιμοποιήσαμε τα 80 δημοφιλέστερα έργα JS από το αποθετήριο GitHub και έγινε ανάλυση σε πάνω από 2000 εκδόσεις αυτών των έργων.

1.2 Οργάνωση του Τόμου

Η παρακάτω μελέτη διαμορφώνεται ως εξής:

Η Ενότητα 2 στοχεύει με την βοήθεια της κατάλληλης βιβλιογραφίας την ανάλυση και επεξήγηση του θεωρητικού υποβάθρου.

Στην Ενότητα 3 παρουσιάζεται το ερευνητικό κομμάτι της μελέτης καθώς και τα ερωτήματα που προκύπτουν

Η Ενότητα 4 περιέχει τα αποτελέσματα και τα στατιστικά στοιχεία της έρευνας

Η Ενότητα 5 συμπεριλαμβάνει τα συμπεράσματα που προέκυψαν από την μελέτη καθώς και τις μελλοντικές επεκτάσεις αυτής.

Στην Ενότητα 6 αναφέρονται οι πιθανές απειλές, η εγκυρότητα και η αξιοπιστία της συγκεκριμένης έρευνας.

Τέλος, στα παραρτήματα βρίσκεται ο κώδικας που χρησιμοποιήθηκε για την συλλογή των δεδομένων

Κεφάλαιο 2: Θεωρητικό Υπόβαθρο

Σε αυτήν την ενότητα θα παρουσιάσουμε το θεωρητικό υπόβαθρο που είναι απαραίτητο για την πλήρη κατανόηση ορισμένων εννοιών τις οποίες ενδεχομένως να μην γνωρίζει ο αναγνώστης. Το εγχείρημα αυτό θα γίνει μέσω της ανάλυσης και επεξήγησης της αρμόζουσας βιβλιογραφίας για το παρών θέμα. Στόχος αυτής της ενότητας είναι ο αναγνώστης να είναι ικανός με βάση τις γνώσεις που απέκτησε από την υποκείμενη βιβλιογραφία να κρίνει και να αξιολογήσει το παρών αντικείμενο χωρίς να χρειάζεται περαιτέρω εμβάθυνση. Σε αυτό το κεφάλαιο θα βρει κανείς στην ενότητα 2.1 τον ορισμό της επαναχρησιμοποίησης λογισμικού, στην ενότητα 2.2 τον ορισμό της εξέλιξης του λογισμικού ενώ στην ενότητα 2.3 και 2.4 τι είναι το GitHub και γιατί έργα JavaScript αντίστοιχα.

2.1 Επαναχρησιμοποίηση Λογισμικού

2.1.1 Ορισμός και Οφέλη που Πηγάζουν από την Επαναχρησιμοποίηση Λογισμικού

Η επαναχρησιμοποίηση λογισμικού είναι μια έννοια η οποία θα μας απασχολήσει σε αρκετές περιπτώσεις και αποτελεί αναμφίβολα σημαντικό κομμάτι της παρούσας διπλωματικής εργασίας οπότε είναι σχεδόν αναγκαίο να γίνει μια πλήρης και εμπεριστατωμένη ανάλυση στο τι ακριβώς είναι και γιατί θα μας απασχολήσει παρακάτω. Ό όρος επαναχρησιμοποίηση λογισμικού μπορεί να ορισθεί με διάφορους τρόπους και ο καθένας μπορεί να δώσει την δικιά του ερμηνεία. Ένας από τους ορισμούς σύμφωνα με τον [2] «**είναι η χρήση υπάρχοντα λογισμικού για την ανάπτυξη νέου λογισμικού αποφεύγοντας την δημιουργία του από την αρχή**» Παράλληλα αποτελεί εργαλείο για σημαντική μείωση στον χρόνο και στο κόστος και θεωρείται από πολλούς ειδικούς επί του θέματος ότι συμβάλλει στην καλύτερη ποιότητα και παραγωγικότητα στην εξέλιξη του λογισμικού. Τα σημαντικότερα πλεονεκτήματα της επαναχρησιμοποίησης λογισμικού είναι :

- **Αύξηση της αξιοπιστίας(Increased Dependability):**Αυτό συμβαίνει καθώς το αρχικό λογισμικό έχει δοκιμαστεί σε συστήματα και παρατηρώντας τα λάθη που προέκυψαν οι μηχανικοί είναι σε θέση ενδεχομένως να τα μειώσουν και να τα διορθώσουν με αποτέλεσμα να δημιουργείται ένα καινούριο πιο αξιόπιστο προϊόν.

- **Αύξηση Παραγωγικότητας(Increased Productivity):** Η χρήση των υπαρχόντων συστατικών ενός λογισμικού είναι πλέον αποδεδειγμένο ότι αποτελεί κλειδί για την αύξηση της παραγωγικότητας και την μείωση του κόστους ανάπτυξης.

- **Αύξηση Αποτελεσματικότητας(Increased Effectiveness):** Ένα επιτυχημένο λογισμικό που βασίζεται στην επαναχρησιμοποίηση είναι αποτελεσματικότερο διότι με την συνεχή δοκιμή στα συστήματα και την πάροδο του χρόνου καταλήγει να εξαφανίσει όποια λάθη είχαν δημιουργηθεί εξ αρχής.

2.1.2 Παράγοντες που Επηρεάζουν την Επαναχρησιμοποίηση Λογισμικού

Όπως αναφέραμε προηγουμένως η επαναχρησιμοποίηση λογισμικού είναι το κύριο χαρακτηριστικό που δίνει την δυνατότητα στο λογισμικό να επαναχρησιμοποιηθεί, λογικό είναι λοιπόν να δημιουργείται η ανάγκη για αξιολόγηση του λογισμικού πριν την χρήση του. Σύμφωνα με τον [3] υπάρχουν 9 παράγοντες που εμπλέκονται στην αξιολόγηση της επαναχρησιμοποίησης του λογισμικού.

- Ευελιξία(Flexibility): Ευελιξία είναι η ικανότητα του λογισμικού να είναι χρήσιμο σε διαφορετικούς σχηματισμούς χωρίς να δεσμεύει τους προγραμματιστές στις διάφορες αλλαγές του συστήματος με την πάροδο του χρόνου.

- Συντηρησιμότητα(Maintainability): Η συντηρησιμότητα είναι η ιδιότητα να δουλεύει σε διαφορετικά συστήματα καθώς είναι αρκετά χρονοβόρο και δύσκολο οι προγραμματιστές να βρουν και να επιλύσουν τα bugs που δημιουργούνται με την αλλαγή του συστήματος.

- Φορητότητα(Portability): Αυτός ο παράγοντας βοηθά κυρίως στην μείωση του κόστους διότι δεν είναι απαραίτητη η εγκατάσταση τρίτου λογισμικού για την χρήση του επαναχρησιμοποιούμενου λογισμικού.

- Κάλυψη Πεδίου(Scope Coverage): Οι προγραμματιστές τείνουν να επιλέγουν ένα λογισμικό για επαναχρησιμοποίηση με μεγαλύτερη κάλυψη αναγκών εάν θέλουν στο μέλλον να επεκτείνουν τα χαρακτηριστικά του λογισμικού τους.

- Σταθερότητα(Stability): Προσδιορίζει το πόσο σταθερό και χωρίς λάθη είναι ένα λογισμικό το οποίο πρόκειται να επαναχρησιμοποιηθεί.

- Κατανοησημότητα(Understandability): Η ιδιότητα του λογισμικού να είναι εύκολα κατανοητό ώστε να μπορούν οι προγραμματιστές να το ενσωματώσουν με πλήρη χρησιμότητα.

- Ιστορικό Χρήσης(Usage History): Το ιστορικό χρήσης φανερώνει το πόσο συχνά έχει χρησιμοποιηθεί το λογισμικό, τις αλλαγές που χρειαστήκαν και γενικότερα δίνει μια ευρύτερη εικόνα για το λογισμικό.

- Μεταβλητότητα(Variability): Η μεταβλητότητα σύμφωνα με τον συγγραφέα αποτελεί δίκοπο μαχαίρι δηλαδή σε ορισμένες περιπτώσεις είναι πλεονέκτημα σε άλλες ίσως όχι.

- Τεκμηρίωση(Documentation): Εάν για οποιονδήποτε λόγο οι οδηγίες χρήσης δεν είναι κατανοητές τότε είναι αρκετά δύσκολο οι μηχανικοί να καταλάβουν το λογισμικό και να το ενσωματώσουν.

Παράλληλα, σύμφωνα με τον [4] υπάρχουν ακόμα πολλά τεχνικά ζητήματα τα οποία αποτρέπουν την επαναχρησιμοποίηση λογισμικού να γίνει αναπόσπαστο κομμάτι διότι οι προγραμματιστές δεν μπορούν να χρησιμοποιήσουν το λογισμικό εάν δεν είναι κυριολεκτικά αλάνθαστος ο κώδικας. Εν κατακλείδι οι απόψεις διίστανται για την χρησιμότητα της επαναχρησιμοποίησης και είναι ένα θέμα που θέλει περαιτέρω συζήτηση.

2.2 Εξέλιξη Λογισμικού

2.2.1 Ορισμός Εξέλιξης Λογισμικού

Αρχικά εξέλιξη λογισμικού είναι η συνεχής ανάπτυξη ενός λογισμικού μετά την αρχική του έκδοση με σκοπό να συμβαδίζει με τις ανάγκες της αγοράς. Η εξέλιξη λογισμικού είναι απαραίτητη διότι οι εταιρίες επενδύουν πολλά χρήματα και είναι πλήρως εξαρτώμενες από το λογισμικό. Σε αυτό το σημείο πρέπει να διαχωρίσουμε την έννοια συντήρηση λογισμικού και εξέλιξη λογισμικού. Η συντήρηση λογισμικού είναι διορθώσεις σε τυχόν bugs και μικρό αλλαγές για την βελτίωση του λογισμικού. Αντίθετα η εξέλιξη στοχεύει περισσότερο στην προσαρμογή και στην μετανάστευση των συστημάτων. Εδώ προκύπτει το εξής ερώτημα, από που γεννιέται η ανάγκη για εξέλιξη λογισμικού; Η ανάγκη αυτή δημιουργείται καθώς τα υπάρχοντα συστήματα δεν θα είναι ποτέ ολοκληρωμένα και συνεχώς θα εξελίσσονται και καθώς εξελίσσονται μεγαλώνει η πολυπλοκότητα τους εκτός εάν βρεθεί μια καλύτερη λύση για τα προβλήματα. Παράλληλα στόχος της εξέλιξης λογισμικού είναι να εξασφαλιστεί η ευελιξία και η αξιοπιστία του συστήματος σε συνάρτηση με την ραγδαία αύξηση στην ζήτηση που υπάρχει τα τελευταία χρόνια.

2.2.2 Οι Νόμοι του Lehman

Την δεκαετία του 1970 ο Manny Lehman πρότεινε ορισμένους νόμους για την εξέλιξη του λογισμικού. Αρχικά χώρισε τα συστήματα που μπορούν αυτοί οι νόμοι να εφαρμοστούν σε 3 κατηγορίες

1. S-type
2. P-type
3. E-type

Τα συστήματα E-type είναι αυτά που απασχολούν τους προγραμματιστές και υπάρχουν στον πραγματικό κόσμο. Συστήματα δηλαδή που δεν υπάρχει σωστή και λάθος λύση καθώς αντικατοπτρίζουν ανθρώπινες συμπεριφορές, επιχειρηματικές αποφάσεις και πολλά άλλα. Τα συστήματα αυτά διαρκώς αλλάζουν σύμφωνα με τις ανάγκες της αγοράς και της ζήτησης. Σύμφωνα με τα παραπάνω ο Manny Lehman προσπάθησε με τον δικό του τρόπο να δώσει μια εξήγηση για αυτά τα συστήματα δημιουργώντας τους παρακάτω νόμους

Πίνακας 1-Νόμοι του Lehman

Ημερομηνία	Τίτλος	Νόμος
1974	**Συνεχής Αλλαγή(Continuing Change)**	Ένα E-type σύστημα πρέπει διαρκώς να προσαρμόζεται αλλιώς φθίνει η χρησιμότητα του μέχρι το σημείο που είναι καλύτερο να κατασκευαστεί από την αρχή.
1974	**Αυξανόμενη Πολυπλοκότητα(Increasing Complexity)**	Κατά την διάρκεια εξέλιξης ενός συστήματος η πολυπλοκότητα του αυξάνεται εάν δεν προσπαθήσει κάποιος να το διατηρήσει ή να το μειώσει
1974	**Αυτορρύθμιση(Self-Regulation)**	Ένα E-type σύστημα είναι μια αυτορρυθμιζόμενη διαδικασία στην οποία τα προϊόντα και οι διαδικασίες πλησιάζουν την κανονική κατανομή
1980	**Διατήρηση οργανωτικής σταθερότητας(Conservation of organizational Stability)**	Ο ρυθμός ανάπτυξης τείνει να είναι σταθερός, εάν δεν εφαρμοστούν κατάλληλοι μηχανισμοί για ένα E-type σύστημα
1980	**Διατήρηση οικειότητας(Conservation of Familiarity)**	Καθώς ένα σύστημα εξελίσσεται όλοι όσοι εμπλέκονται σε αυτό προγραμματιστές, πωλητές χρήστες κ.α. πρέπει να παραμένουν οικείοι με το περιεχόμενο διότι το σύστημα δεν μπορεί να έχει τεράστιες αλλαγές μεταξύ των εκδόσεων και οι αλλαγές να εξελίσσονται με σταθερό ρυθμό ώστε το σύστημα να είναι παραγωγικό
1980	**Συνεχής Ανάπτυξη(Continuing Growth)**	Οι λειτουργίες του συστήματος πρέπει διαρκώς να αυξάνονται σε όλη την διάρκεια ζωής του με σκοπό την ικανοποίηση των χρηστών
1996	**Φθίνουσα ποιότητα(Declining Quality)**	Η ποιότητα ενός συστήματος τείνει να μειώνεται εκτός εάν προσαρμόζεται στις αλλαγές του περιβάλλοντος
1996	**Ανατροφοδοτούμενο Σύστημα(Feedback System)**	Η διαδικασία ανάπτυξης συστημάτων πρέπει να αντιμετωπίζεται ως ένα ανατροφοδοτούμενο σύστημα πολλών επιπέδων, αναδράσεων και εμπλεκομένων

2.3 Εξέλιξη και Επαναχρησιμοποίηση Λογισμικού

Στις παραπάνω δύο ενότητες εξηγήσαμε τις έννοιες επαναχρησιμοποίηση και εξέλιξη λογισμικού, σε αυτήν την ενότητα θα διαπιστώσουμε ότι αυτές οι δύο έννοιες είναι αλληλεξαρτώμενες καθώς δεν υφίσταται επαναχρησιμοποίηση χωρίς εξέλιξη. Σύμφωνα με την μελέτη [5] η επαναχρησιμοποίηση λογισμικού είναι μια διαδικασία εξέλιξης διότι οι προγραμματιστές έχουν την τάση να επαναχρησιμοποιούν καινούρια αντικείμενα κατά την διάρκεια της εξέλιξης του έργου αλλά και να αναβαθμίζουν τα υπάρχοντα επαναχρησιμοποιούμενα αντικείμενα με σκοπό να μειώσουν σημαντικά την εμφάνιση λαθών τα οποία τείνουν να είναι λιγότερα στις τελευταίες εκδόσεις ενός επαναχρησιμοποιήσιμου αντικειμένου. Ωστόσο οι μεγάλες αλλαγές στο δομικό κομμάτι ενός έργου το οποίο επαναχρησιμοποιείται επηρεάζει σημαντικά τον χρόνο και τον κόπο που απαιτείται για την αναβάθμιση των αντικειμένων στα έργα που χρησιμοποιούν το επαναχρησιμοποιούμενο έργο, με αποτέλεσμα η αναβάθμιση των αντικειμένων ενός έργου να είναι άμεσα συνδεδεμένη με την εξέλιξη του επαναχρησιμοποιούμενου έργου. Ακόμη καθώς ένα επαναχρησιμοποιούμενο έργο εξελίσσεται αποκτά καινούριες λειτουργίες ή γίνονται προσπάθειες για συντήρηση όπως για παράδειγμα την βελτίωση της δομής του κώδικα ή την επίλυση λαθών, αυτές οι αλλαγές προκαλούν αρνητικές συνέπειες στα άλλα έργα διότι πρέπει να γίνει ανακατασκευή του κώδικα. Παρόλα αυτά η εξέλιξη ενός έργου παράγει πολλά οφέλη τα οποία μεταφέρονται και στα άλλα έργα, οφέλη τα οποία έκαναν την επαναχρησιμοποίηση λογισμικού καθώς και την εξέλιξη λογισμικού τόσο δημοφιλείς πρακτικές στις μέρες μας.

2.4 Αποθετήριο GitHub

Το GitHub είναι μία ιστοσελίδα και παράλληλα μια cloud-based υπηρεσία η οποία βοηθά τους προγραμματιστές να αποθηκεύσουν, να επιβλέπουν, να παρακολουθούν και να ελέγχουν τις αλλαγές στον κώδικα τους. Για να καταλάβουμε πραγματικά τι ακριβώς είναι το GitHub αρκεί να εξηγήσουμε τις παρακάτω έννοιες :

- Έλεγχος Εκδόσεων (Version Control): Επιτρέπει στους προγραμματιστές ή την ομάδα προγραμματιστών να παίρνουν ένα κομμάτι του κώδικα με ασφάλεια και να εφαρμόζουν τις απαραίτητες αλλαγές χωρίς να επηρεάζεται το υπόλοιπο έργο. Μόλις υλοποιηθούν αυτές οι αλλαγές και ο κώδικας είναι λειτουργικός ενσωματώνεται στο υπόλοιπο κομμάτι χωρίς να δημιουργείται κάποιο πρόβλημα. Αυτή η τεχνική είναι γνωστή ως branching and merging

- Git: Το Git είναι ένα σύστημα ελέγχου έκδοσης ανοιχτού κώδικα που ξεκίνησε από τον Linus Torvalds τον ιδρυτή του Linux. Με απλά λόγια όταν ένας προγραμματιστής δημιουργεί ένα έργο και στην συνέχεια πραγματοποιεί μεταρρυθμίσεις στον κώδικα και απελευθερώνει νέες εκδόσεις το Git του επιτρέπει να αποθηκεύσει αυτές τις αλλαγές σε ένα κεντρικό αποθετήριο στο οποίο κάθε προγραμματιστής μπορεί να συνεισφέρει κατεβάζοντας τις νέες εκδόσεις και κάνοντας αλλαγές.

Συνδυάζοντας τα παραπάνω το GitHub είναι μια ιστοσελίδα η οποία έχει ως βάση το git και γενικότερα τον έλεγχο εκδόσεων μέσα από ένα πολύ φιλικό περιβάλλον αλληλεπίδρασης και μπορεί ο οποιοσδήποτε να κάνει εγγραφή και να ανεβάσει το δικό του υλικό ή να βοηθήσει άλλους εντελώς δωρεάν. Ακόμη το GitHub αποτελεί ένα μέσο κοινωνικής δικτύωσης καθώς κάθε χρήστης έχει το δικό του προφίλ που λειτουργεί σαν βιογραφικό παρουσιάζοντας προηγούμενα έργα του αλλά και την συνεισφορά του σε έργα τρίτων. Τέλος το GitHub δεν είναι ένα μέρος μόνο για προγραμματιστές καθώς υπάρχει ένα μικρό ποσοστό το οποίο χρησιμοποιεί το GitHub για να ανεβάσει οποιοδήποτε είδος αρχείου χωρίς περιορισμούς.

2.5 Έργα JavaScript

Γλώσσα προγραμματισμού είναι ο τρόπος με τον οποίο ο άνθρωπος μπορεί και αλληλοεπιδρά με τον υπολογιστή, μία από αυτές τις γλώσσες είναι και η JavaScript. Οι διασημότερες σελίδες κοινωνικής δικτύωσης Facebook,Twitter αλλά και σελίδες αγοράς προϊόντων όπως η Amazon βασίζονται κυρίως σε JavaScript καθώς λειτουργεί σε όλα τα προγράμματα περιήγησης πράγμα που την καθιστά ίσως την δημοφιλέστερη γλώσσα αυτήν την στιγμή. Σε έρευνα της Stack Overflow την οποία περιέγραψαν ως *«η μεγαλύτερη προγραμματιστική ερεύνα που διεξάχθηκε»* η JavaScript ανακηρύχθηκε η δημοφιλέστερη γλώσσα προγραμματισμού *«Για πέμπτη συνεχόμενη χρονιά είναι η γλώσσα την οποία χρησιμοποιούν οι περισσότεροι προγραμματιστές»* σύμφωνα με την ίδια έρευνα. Η JavaScript είναι μία γλώσσα η οποία συνεχώς μεγαλώνει και αποτελεί εργαλείο για την ανάπτυξη πολύπλοκων προγραμμάτων και είναι αρκετά ευέλικτη και χρήσιμη για ανάπτυξη ιστοσελίδας με στοιχεία αλληλεπίδρασης που λειτουργούν σωστά για τον χρήστη και είναι ελκυστικά στο μάτι. Επίσης ο συνδυασμός αντικειμενοστραφούς και δομημένου προγραμματισμού που προσφέρει αποτελεί σημαντικό κίνητρο και κεντρίζει το ενδιαφέρον των προγραμματιστών αφού τους παρέχει περισσότερη ελευθερία και μεγαλύτερο περιθώριο να προγραμματίσουν με οποίον τρόπο κρίνουν σωστότερο. Εν κατακλείδι οι προγραμματιστικές δυνατότητες που προσφέρει η JavaScript σε συνδυασμό με τον βαθμό που είναι διαδεδομένη στον προγραμματιστικό κόσμο αποτελεί εξαιρετικό υπόβαθρο για να μελετήσουμε την εξέλιξη των επαναχρησιμοποιούμενων έργων JavaScript.

Κεφάλαιο 3: Μετρικές και η Εφαρμογή τους

Σε αυτό το κεφάλαιο θα αναλύσουμε το ερευνητικό κομμάτι αυτής της μελέτης, τα ερωτήματα που προκύπτουν και τους τρόπους με τους οποίους θα τα προσεγγίσουμε.

3.1 Ορισμός Μετρικών Λογισμικού

Σύμφωνα με τους [6] **«οι μετρικές λογισμικού παρέχουν στα χαρακτηριστικά του λογισμικού ορισμένες ποσοτικές περιγραφές, χαρακτηριστικά τα οποία εξάγονται από το προϊόν του λογισμικού, την διαδικασία ανάπτυξης του λογισμικού και τους αντίστοιχους πόρους.»**

- Προϊόν Λογισμικού: Τα αρχεία και τα προγράμματα που παράγονται κατά την ανάπτυξη του λογισμικού.

- Διαδικασία Ανάπτυξης: Διάφορες διαδικασίες σχετικά με την ανάπτυξη του λογισμικού όπως πχ σχεδίαση λογισμικού, συντήρηση και δοκιμές.

- Πόροι: Οι Προγραμματιστές, το κόστος του προϊόντος και της διαδικασίας.

Μια άλλη προσέγγιση για να γίνει περισσότερο κατανοητό τι ακριβώς είναι η μετρική λογισμικού, σύμφωνα με τον [7] **«μετρική είναι μια συνάρτηση η οποία δέχεται σαν είσοδο τα δεδομένα του λογισμικού και έχει ως έξοδο μία τιμή η οποία αποφασίζει κατά πόσο αυτά τα δεδομένα επηρεάζουν το λογισμικό.»** Οι μετρικές λογισμικού χωρίζονται σε τρείς διαφορετικές κατηγορίες

1. Μετρικές Διαδικασίας: Οι συγκεκριμένες μετρικές επικεντρώνονται κυρίως στο χρόνο που θα διαρκέσει η ανάπτυξη του λογισμικού, το κόστος για την ανάπτυξη και κατά πόσο αποτελεσματικοί είναι οι μέθοδοι που χρησιμοποιούνται σε σχέση με ενναλακτικές μεθόδους. Περιλαμβάνει την βελτίωση της διαδικασίας και μελλοντικών διαδικασιών. Αυτές οι μετρικές είναι απαραίτητες για να γίνεται έλεγχος και διαχείριση όλης της διαδικασίας ανάπτυξης ενός λογισμικού

2. Μετρικές Έργου: Μετρικές για την κατανόηση και τον έλεγχο της κατάστασης που βρίσκεται το έργο. Περιέχουν κυρίως κλίμακες, φόρτο εργασίας, το ρίσκο που δημιουργείται και τον βαθμό της ανταπόκρισης των πελατών. Σκοπός αυτών είναι να αναβαθμίσουνε την ποιότητα του λογισμικού μέσω προηγμένων τεχνικών και στρατηγικών διαχείρισης.

3. Μετρικές Προϊόντος: Είναι υπεύθυνες για τον έλεγχο της αξιοπιστίας, συντηρησιμότητας, φορητότητας και της πολυπλοκότητας του λογισμικού. Εν τέλει οι μετρικές είναι απαραίτητες για την κατανόηση έργου, της διαδικασίας ανάπτυξης, συμβάλλουν στην αξιολόγηση της συμπεριφοράς του λογισμικού και προβλέπουν τα προβλήματα που μπορούν να δημιουργηθούν για το συγκεκριμένο έργο άλλα και μελλοντικά έργα.

3.2 Μετρικές που Εφαρμόστηκαν

Σε αυτήν την ενότητα θα αναλύσουμε τις μετρικές που χρησιμοποιήθηκαν για να εξετάσουμε την αποτελεσματικότητα των νόμων του Lehman για την εξέλιξη του λογισμικού σε επαναχρησιμοποιούμενα έργα JavaScript από το αποθετήριο GitHub.

Πίνακας 2-Μετρικές βασισμένες στους νόμους του Lehman

Νόμοι του Lehman	**Μεταβλητές**
Νόμος 1 (Continuing Change)	[V_1] Μέρες για την αναβάθμιση των πακέτων
Νόμος 2 (Increasing Complexity)	[V_2] Συνολικός αριθμός των μοναδικών πακέτων για κάθε project
Νόμος 3 (Self-Regulation)	[V_3] Αριθμός καινούριων πακέτων που προστέθηκαν για κάθε project
Νόμος 4 (Conservation of organizational stability)	[$V_{4.1}$] $\frac{Αριθμός\ Πακέτων\ που\ προσθέθηκαν\ και\ αφαιρέθηκαν}{Μέρες\ μεταξύ\ της\ έκδοσης\ project}$ [$V_{4.2}$]Αριθμός μοναδικών πακέτων σε προηγούμενες εκδόσεις [$V_{4.3}$]Αριθμός μοναδικών πακέτων στην τωρινή έκδοση
Νόμος 5(Conservation of familiarity)	[V_5]Συνολικές Αλλαγές = Αριθμός πακέτων που προστέθηκαν + Αριθμό πακέτων που αφαιρέθηκαν
Νόμος 6 (Continuing Growth)	[V_6]Πακέτα που προστέθηκαν και αφαιρέθηκαν σε σχέση με προηγούμενες εκδόσεις
Νόμος 7 (Declining quality)	[V_7]Αριθμός εκδόσεων μεταξύ του πακέτου που χρησιμοποιείται στο project και την τελευταία έκδοση του πακέτου.

3.3 Διαδικασία Εφαρμογής των Μετρικών

Για την εφαρμογή των παραπάνω μετρικών χρησιμοποιήσαμε υλικό από 80 έργα JavaScript από το αποθετήριο GitHub. Τα έργα αυτά δεν επιλέχθηκαν τυχαία καθώς για να χρησιμοποιήσουμε ένα έργο θα έπρεπε ο συγγραφέας να έχει εκδώσει περισσότερες από δυο εκδόσεις και να βρίσκονται στις 50 πρώτες σελίδες όταν εφαρμόζεται το φίλτρο most forks. Ανοίγοντας ένα τέτοιο έργο θα διαπιστώσει κανείς ότι μέσα υπάρχει ένα αρχείο που ονομάζεται package.json. Εκεί μέσα βρίσκεται ένα branch το οποίο περιέχει όλα τα πακέτα devDependencies, τα πακέτα αυτά είναι τα πακέτα που επαναχρησιμοποιεί το συγκεκριμένο έργο. Με βάση τα παραπάνω σκοπός μας είναι να συλλέξουμε αυτά τα επαναχρησιμοποιούμενα πακέτα από κάθε έκδοση του έργου ώστε να εφαρμόσουμε τις παραπάνω μετρικές και να βγάλουμε τα αντίστοιχα συμπεράσματα. Για την συλλογή αυτών των πακέτων δημιουργήθηκε κατάλληλος κώδικας σε γλώσσα bash, διότι είναι ο ιδανικός τρόπος για να αποκτήσει κάποιος μεγάλο όγκο δεδομένων απλά και γρήγορα. Έτσι με τις εντολές κυρίως curl και wget από το API του αποθετηρίου GitHub κατεβάσαμε τις τελευταίες 30 πιο πρόσφατες εκδόσεις από κάθε έργο. Αφού κατεβάσαμε τις εκδόσεις κρατήσαμε μόνο το αρχείο package.json για κάθε έκδοση του έργου. Στην συνέχεια φιλτράραμε το αρχείο package.json για να κρατήσουμε τα πακέτα που βρίσκονται στο branch devDependencies. Τέλος ενώσαμε αυτά τα πακέτα μαζί με την ημερομηνία της έκδοσης σε ένα τελικό αρχείο .txt το οποίο ονομάσαμε packunique(μοναδικά πακέτα) το οποίο είναι ένα αρχείο κειμένου με όλα τα πακέτα που επαναχρησιμοποιούνται στην εκάστοτε έκδοση του project.

async/package.json

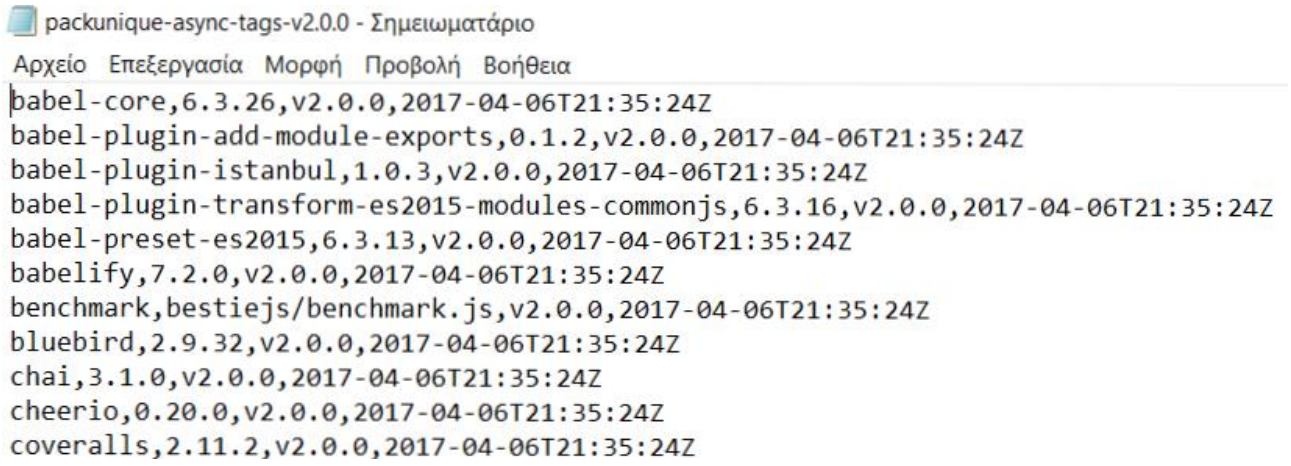
```
"devDependencies": {
"babel-core": "^6.26.3",
"babel-eslint": "^8.2.6",
"babel-minify": "^0.5.0",
"babel-plugin-add-module-exports": "^0.2.1",
"babel-plugin-istanbul": "^5.1.4",
"babel-plugin-syntax-async-generators": "^6.13.0",
"babel-plugin-transform-es2015-modules-commonjs": "^6.26.2",
"babel-preset-es2015": "^6.3.13",
"babel-preset-es2017": "^6.22.0",
"babel-register": "^6.26.0",
"babelify": "^8.0.0",
"benchmark": "^2.1.1",
"bluebird": "^3.4.6",
"browserify": "^16.2.3",
"chai": "^4.2.0",
"cheerio": "^0.22.0",
"coveralls": "^3.0.4",
"eslint": "^6.0.1",
"karma": "^4.1.0",
"mocha": "^6.1.4",
"nyc": "^14.1.1",
```

Σχήμα 1-Αρχείο PACKAGE.JSON

```
[
 {
  "url": "https://api.github.com/repos/caolan/async/releases/6005479",
  "assets_url": "https://api.github.com/repos/caolan/async/releases/6005479/assets",
  "upload_url": "https://uploads.github.com/repos/caolan/async/releases/6005479/assets{?name,label}",
  "html_url": "https://github.com/caolan/async/releases/tag/v2.0.0",
  "id": 6005479,
  "author": {
   "login": "megawac",
   "id": 3475472,
   "node_id": "MDQ6VXNlcjM0NzU0NzI=",
   "avatar_url": "https://avatars.githubusercontent.com/u/3475472?v=4",
   "gravatar_id": "",
   "url": "https://api.github.com/users/megawac",
   "html_url": "https://github.com/megawac",
   "followers_url": "https://api.github.com/users/megawac/followers",
   "following_url": "https://api.github.com/users/megawac/following{/other_user}",
   "gists_url": "https://api.github.com/users/megawac/gists{/gist_id}",
   "starred_url": "https://api.github.com/users/megawac/starred{/owner}{/repo}",
   "subscriptions_url": "https://api.github.com/users/megawac/subscriptions",
   "organizations_url": "https://api.github.com/users/megawac/orgs",
   "repos_url": "https://api.github.com/users/megawac/repos",
   "events_url": "https://api.github.com/users/megawac/events{/privacy}",
   "received_events_url": "https://api.github.com/users/megawac/received_events",
"type": "User",
   "site_admin": false
```


Σχήμα 2-API GitHub

έργο caolan/async στο API του αποθετηρίου GitHub



Εικόνα 1-Packunique Αρχείο

Στην εικόνα 1 φαίνεται ένα αρχείο packunique.txt με πρώτη στήλη το όνομα του πακέτου, δεύτερη στήλη την έκδοση του πακέτου, τρίτη στήλη την έκδοση του έργου και τελευταία στήλη την ημερομηνία της έκδοσης του έργου async. Διαχωριστικός παράγοντας μεταξύ των στηλών είναι το “ , ”

Μετρική 1 - Μέρες για την αναβάθμιση των πακέτων

Για αυτήν την μετρική συγκρίναμε τα επαναχρησιμοποιούμενα πακέτα δύο διαδοχικών εκδόσεων και με την βοήθεια από το API του npm.js δημιουργήσαμε τελικά ένα αρχείο .txt με όλα τα επαναχρησιμοποιούμενα πακέτα, την ημερομηνία έκδοσης των πακέτων αυτών και την ημερομηνία έκδοσης της έκδοσης. Για να διαπιστώσουμε πόσο συχνά γίνεται αναβάθμιση των πακέτων αφαιρέσαμε την ημερομηνία της έκδοσης με την ημερομηνία που εκδόθηκε το πακέτο και δημιουργήσαμε ένα τελικό .txt. Παρακάτω φαίνεται η εικόνα πως είναι διαμορφωμένα τα αρχεία αυτά.

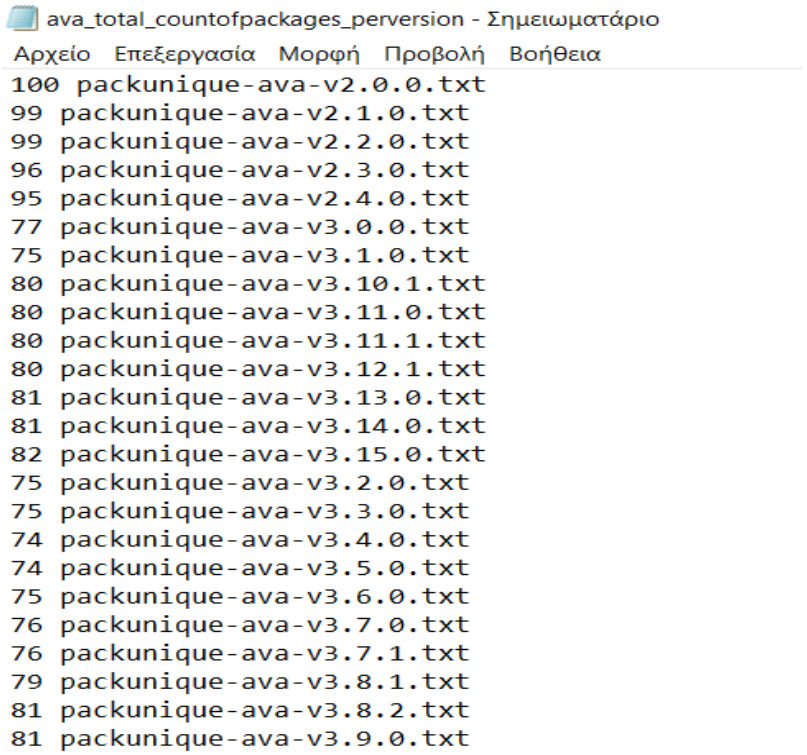
```
Package,Package Version,Package Date,Project Version and Date, Days Between Package Update:
babel-core,6.3.26,2015-12-23,v2.0.0,2017-04-06T21:35:24Z,2 years and 260 days
babel-plugin-add-module-exports,0.1.2,2015-12-18,v2.0.0,2017-04-06T21:35:24Z,2 years and 255 days
babel-plugin-istanbul,1.0.3,2016-07-09,v2.0.0,2017-04-06T21:35:24Z,1 years and 94 days
babel-plugin-transform-es2015-modules-commonjs,6.3.16,2015-12-09,v2.0.0,2017-04-06T21:35:24Z,2 years and 246 days
babel-preset-es2015,6.3.13,2015-12-04,v2.0.0,2017-04-06T21:35:24Z,2 years and 241 days
babelify,7.2.0,2015-11-02,v2.0.0,2017-04-06T21:35:24Z,2 years and 209 days
benchmark,2.1.1,2016-07-20,v2.1.0,2017-04-06T21:34:21Z,1 years and 105 days
bluebird,2.9.32,2015-07-03,v2.0.0,2017-04-06T21:35:24Z,2 years and 87 days
chai,3.1.0,2015-07-16,v2.0.0,2017-04-06T21:35:24Z,2 years and 100 days
cheerio,0.20.0,2016-02-01,v2.0.0,2017-04-06T21:35:24Z,1 years and 65 days
coveralls,2.11.2,2014-09-22,v2.0.0,2017-04-06T21:35:24Z,3 years and 168 days
```

Εικόνα 2-Μετρική 1 Αρχεία

Στην εικόνα 2 φαίνεται το αρχείο της μετρικής 1 το οποίο είναι χωρισμένο με “ , “ σε στήλες. Πρώτη στήλη είναι το όνομα του πακέτου, δεύτερη στήλη η έκδοση του πακέτου, τρίτη στήλη η ημερομηνία της έκδοσης του πακέτου, τέταρτη στήλη η έκδοση και η ημερομηνία του έργου και τελευταία στήλη η αφαίρεση των δύο ημερομηνιών.

Μετρική 2 - Συνολικός αριθμός των μοναδικών πακέτων για κάθε project

Στην μετρική 2 μετρήσαμε πόσα πακέτα περιέχει κάθε έκδοση του έργου μετρώντας τις σειρές που υπάρχουν στα αρχεία packunique.txt. Αυτή η διαδικασία έγινε για όλες τις εκδόσεις του έργου και τις ενώσαμε σε ένα τελικό αρχείο .txt, το οποίο απαριθμεί τον συνολικό αριθμό μοναδικών πακέτων κάθε έργου. Παρακάτω φαίνεται η εικόνα πως είναι διαμορφωμένα τα αρχεία αυτά

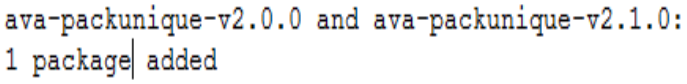


Εικόνα 3-Μετρική 2 Αρχεία

Στην εικόνα 3 φαίνεται ένα αρχείο της μετρικής 2 στο οποίο εμπεριέχεται ο αριθμός των πακέτων για κάθε έκδοση του έργου ava.

Μετρική 3 - Αριθμός καινούριων πακέτων που προστέθηκαν για κάθε project

Στην μετρική 3 συγκρίναμε δύο διαδοχικές εκδόσεις κάθε έργου με σκοπό να δούμε πόσα και ποια πακέτα προστέθηκαν κάθε φορά που ο συγγραφέας εκδίδει μια καινούρια έκδοση. Παρακάτω φαίνεται η εικόνα πως είναι διαμορφωμένα τα αρχεία αυτά.



Εικόνα 4-Μετρική 3 Αρχεία

Στην εικόνα 4 φαίνεται ένα αρχείο της μετρικής 3 το οποίο περιέχει τα πακέτα που προστέθηκαν μεταξύ δύο διαδοχικών εκδόσεων v2.0.0 και v2.1.0 του έργου ava.

Μετρική 4.1 -Διαίρεση με αριθμητή τον αριθμό των πακέτων που προστέθηκαν και αφαιρέθηκαν και παρονομαστή τον αριθμό των ημερών μεταξύ δύο διαδοχικών εκδόσεων

Σε αυτήν την μετρική συγκρίναμε δύο διαδοχικές εκδόσεις και κρατήσαμε το σύνολο των πακέτων που προστέθηκαν και αφαιρέθηκαν μεταξύ των δύο εκδόσεων. Στην συνέχεια από τα αρχεία packunique πήραμε τις ημερομηνίες των δύο εκδόσεων και τις αφαιρέσαμε για να βρούμε πόσες μέρες χρειάστηκαν για να δημοσιεύσει ο συγγραφέας την νέα έκδοση. Τέλος διαιρέσαμε τους δύο αυτούς αριθμούς. Στην παρακάτω εικόνα φαίνεται πώς έχουν διαμορφωθεί τα αρχεία αυτά.

```
ava-v2.0.0 and ava-v2.1.0:
0.33
```

Εικόνα 5-Μετρική 4.1 Αρχεία

Στην εικόνα 5 φαίνεται μεταξύ δύο διαδοχικών εκδόσεων v2.0.0 και v2.1.0 του έργου ava το αποτέλεσμα της διαίρεσης που αναφέρθηκε προηγουμένως.

Μετρική 4.2 – Αριθμός μοναδικών πακέτων σε προηγούμενες εκδόσεις

Για την μετρική 4.2 δεν δημιουργήθηκαν εκ νέου αρχεία καθώς περιέχονται ήδη στα αρχεία της μετρικής 2 οπότε θεωρήσαμε περιττό την δημιουργία ξεχωριστών αρχείων

Μετρική 4.3 – Αριθμός μοναδικών πακέτων στην τωρινή έκδοση

Όπως και στην παραπάνω μετρική έτσι και σε αυτήν δεν δημιουργήθηκαν ξεχωριστά αρχεία.

Οπότε επιλέξαμε αυτές οι μετρικές να μην αναλυθούν ξεχωριστά αλλά η μία σε σχέση με την άλλη. Ουσιαστικά θα συγκρίνουμε των αριθμό των μοναδικών πακέτων στην τωρινή έκδοση σε σχέση με την πρώτη έκδοση για να αξιολογήσουμε την εξέλιξη που είχε το έργο.

Μετρική 5 – Αριθμός των πακέτων που προστέθηκαν και αριθμός των πακέτων που αφαιρέθηκαν

Στην μετρική 5 συγκρίναμε δύο διαδοχικές εκδόσεις και βρήκαμε πόσες εκδόσεις προστέθηκαν και πόσες αφαιρέθηκαν και προσθέσαμε αυτόν τον αριθμό για να καταλήξουμε στις συνολικές αλλαγές που έγιναν μεταξύ των δύο διαδοχικών εκδόσεων. Στην παρακάτω εικόνα φαίνεται πως διαμορφώνονται τα αρχεία αυτά.

```
 ava-v2.0.0 and ava-v2.1.0
2 total changes
```

Εικόνα 6-Μετρική 5 Αρχεία

Στην εικόνα 6 φαίνεται ένα αρχείο της μετρικής 5 το οποίο περιέχει τις συνολικές αλλαγές που έγιναν μεταξύ δύο διαδοχικών εκδόσεων v2.0.0 και v2.1.0 του έργου ava.

Μετρική 6 - Πακέτα που προστέθηκαν και αφαιρέθηκαν σε σχέση με προηγούμενες εκδόσεις

Η διαφορά αυτής της μετρικής με την μετρική 3 είναι ότι σε αυτήν την μετρική δεν μας ενδιαφέρουν πόσα πακέτα προστέθηκαν αλλά ποια πακέτα προστέθηκαν και αφαιρέθηκαν. Έτσι συγκρίναμε δύο διαδοχικές εκδόσεις για να δούμε τις διαφορές μεταξύ των δύο εκδόσεων. Παρακάτω φαίνεται η εικόνα με το πως έχουν διαμορφωθεί τα αρχεία αυτά.

```
ava-v2.0.0 and ava-v2.1.0
removed
 nyc
**********************************
added
@types/node
```

Εικόνα 7-Μετρική 6 Αρχεία

Σε αυτό το σημείο φαίνεται ξεκάθαρα ότι για το έργο ava και τις διαδοχικές εκδόσεις v2.0.0 και v2.1.0 τα αποτελέσματα των μετρικών 3,5,6 ταυτίζονται καθώς στην μετρική 3 προστέθηκε 1 πακέτο το @types/node και αφαιρέθηκε το nyc άρα δύο συνολικές αλλαγές όπως δείχνει και η μετρική 5.

Μετρική 7 - Αριθμός εκδόσεων μεταξύ του πακέτου που χρησιμοποιείται στο project και την τελευταία έκδοση του πακέτου.

Στην μετρική 7 πήραμε την έκδοση κάθε πακέτου από τα αρχεία packunique και την συγκρίναμε με την τελευταία έκδοση που δημοσιεύθηκε στο npm.js. Με αυτόν τον τρόπο μπορέσαμε να δούμε πόσες εκδόσεις πίσω βρίσκονται τα επαναχρησιμοποιούμενα πακέτα που υπάρχουν σε κάθε έργο. Παρακάτω φαίνεται η εικόνα του πως είναι διαμορφωμένα τα αρχεία αυτά.

```
Package name,Current Versions,Used Version,Project name,Project Version,Number of versions behind:
babel-core,7.0.0-ph.9,6.3.26,async,v2.0.0,40
babel-plugin-add-module-exports,1.0.4,0.1.2,async,v2.0.0,14
babel-plugin-istanbul,6.1.1,1.0.3,async,v2.0.0,31
babel-plugin-transform-es2015-modules-commonjs,7.0.0-ph.9,6.3.16,async,v2.0.0,26
babel-preset-es2015,7.0.0-ph.9,6.3.13,async,v2.0.0,16
babelify,9.0.0,7.2.0,async,v2.0.0,3
benchmark,,bestiejs/benchmark.js,async,v2.0.0,-1
bluebird,3.7.2,2.9.32,async,v2.0.0,48
chai,4.3.4,3.1.0,async,v2.0.0,19
cheerio,1.0.0-.9,0.20.0,async,v2.0.0,31
coveralls,3.1.1,2.11.2,async,v2.0.0,37
```

Εικόνα 8-Μετρική 7 Αρχεία

Στην εικόνα 8 φαίνεται ένα αρχείο της μετρικής 7 το οποίο είναι χωρισμένο με " , " σε στήλες. Πρώτη στήλη όνομα πακέτου, δεύτερη στήλη τωρινή έκδοση του πακέτου, τρίτη στήλη η έκδοση του πακέτου που χρησιμοποιείται στο έργο, τέταρτη στήλη το όνομα του έργου και τελευταία στήλη είναι ο αριθμός που αντιπροσωπεύει την διαφορά μεταξύ της τωρινής έκδοσης με την χρησιμοποιούμενη έκδοση.

Είναι αναγκαίο να σημειωθεί ότι οι παραπάνω εικόνες διαμορφώθηκαν με αυτόν τον τρόπο ώστε να βοηθήσουν τον αναγνώστη να καταλάβει καλύτερα την διαδικασία εφαρμογής των μετρικών και μπορεί να απέχουν από το πώς είναι πραγματικά τα αρχεία τα οποία απεικονίζονται.

Ολοκληρώνοντας το κεφάλαιο αυτό ευελπιστούμε ο αναγνώστης να έχει κατανοήσει πλήρως τι είναι μετρική λογισμικού, για ποιόν λόγο είναι τόσο σημαντικές και τις κατηγορίες που υπάρχουν. Παράλληλα ελπίζουμε ότι η παραπάνω ανάλυση της διαδικασίας με την οποία εφαρμόστηκαν οι μετρικές και η χρήση εικόνων θα βοηθήσουν τον αναγνώστη να καταλάβει και να αξιολογήσει τα αποτελέσματα των μετρικών τα οποία θα παρουσιαστούν στο επόμενο κεφάλαιο.

Κεφάλαιο 4: Εξαγωγή Αποτελεσμάτων

4.1 Χρήση του Λογισμικού JASP

Το JASP είναι ένα δωρεάν λογισμικό ανοιχτού κώδικα για στατιστική ανάλυση που υποστηρίζεται από το Πανεπιστήμιο του Amsterdam. Το εύχρηστο και φιλικό περιβάλλον χρήσης που παρέχει σε συνδυασμό με την ικανότητα του να εκτελεί απλές και σύνθετες αναλύσεις το καθιστά επάξιο ανταγωνιστή του SPSS. Γιατί όμως κάποιος να επιλέξει το JASP έναντι της σίγουρης λύσης που παρέχει το SPSS; Σύμφωνα με τον [8] παρόλο που πολλά πανεπιστήμια χρησιμοποιούν κυρίως το SPSS η τιμή του δεν είναι και τόσο προσιτή ειδικά εάν δεν σχετίζεσαι με κάποιο πανεπιστήμιο. Έτσι η δωρεάν παροχή που προσφέρει το JASP είναι αρκετά ελκυστικός παράγοντας για να το προτιμήσει κάποιος. Παρακάτω βρίσκεται μια λίστα με τα πλεονεκτήματα του JASP.

- Μεγέθη αποτελεσμάτων: Σε αντίθεση με το SPSS που προσφέρει έναν περιορισμένο αριθμό μεγεθών, στο JASP σου δίνεται η δυνατότητα να επιλέγεις τα μεγέθη τα οποία κρίνεις ότι είναι απαραίτητα για την μελέτη σου κάνοντας απλά ένα τικ σε οποιοδήποτε μέγεθος θέλεις να προσθέσεις σε κάθε έργο.

- Συνεχής Αναβάθμιση των αποτελεσμάτων: Ίσως ένα από τα σημαντικότερα πλεονεκτήματα του JASP είναι ότι όλες οι επιλογές και τα αποτελέσματα των μεγεθών που επιλέγονται εμφανίζονται σε πραγματικό χρόνο. Αντίθετα στο SPSS εάν έχεις τυπώσει τα αποτελέσματα και συνειδητοποιήσεις ότι ξέχασες να συμπεριλάβεις μια επιλογή ή ένα μέγεθος θα πρέπει να επαναλάβεις όλη την διαδικασία από την αρχή το οποίο είναι αρκετά κουραστικό και χρονοβόρο.

- Μινιμαλιστικός Σχεδιασμός: Η τέχνη του ελαχίστου, στο JASP σε αντίθεση με άλλα λογισμικά σου δίνει την δυνατότητα να ξεκινάς με τα απολύτως βασικά μεγέθη/αποτελέσματα και έχεις την ευχέρεια να προσθέσεις εσύ ότι θεωρείς απαραίτητο και οποιαδήποτε στιγμή επιθυμείς.

- Επαναχρησιμοποίηση των Αναλύσεων: Το JASP σου δίνει την ευκαιρία να αποθηκεύσεις τα αποτελέσματα σου σε αρχείο με μορφή .JASP, οπότε μπορείς να ανοίξεις στο μέλλον αυτό το αρχείο και να εκτελέσεις αλλαγές που θεωρείς σωστές την δεδομένη χρονική στιγμή.

Τέλος, σύμφωνα με τα παραπάνω, είναι δεδομένο ότι το JASP έχει κάνει αισθητή την παρουσία του στον χώρο των λογισμικών στατιστικής ανάλυσης και σίγουρα θα αποτελέσει εργαλείο για πολλούς ερευνητές.

Εικόνα 9-JASP Interface

4.2 Αποτελέσματα Μετρικών

Αποπερατώνοντας την διαδικασία που αναφέρθηκε στο προηγούμενο κεφάλαιο έχουμε συγκεντρώσει έναν σημαντικό όγκο αρχείων που περιέχουν τα δεδομένα τα οποία θα μας βοηθήσουν να καταλήξουμε σε ασφαλή συμπεράσματα. Τα δεδομένα αυτά καθαυτά δεν μπορούν να θεωρηθούν αξιόπιστο εργαλείο για να καταλάβουμε εάν όντως η μετρικές που επιλέξαμε είναι χρήσιμες για να κατανοήσουμε εάν οι νόμοι του Lehman έχουν εφαρμογή στην σημερινή εποχή. Σε αυτό το σημείο είναι που θα χρειαστούμε το λογισμικό JASP του οποίου τα χαρακτηριστικά αναφέρθηκαν στην προηγούμενη ενότητα. Με την υποστήριξη από το JASP μπορούμε να εισάγουμε τα δεδομένα μας και να εξάγουμε στατιστικά μεγέθη για να έχουμε μια ολοκληρωμένη εικόνα από τι αποτελούνται τα δεδομένα μας και τι αντιπροσωπεύουν. Τα μεγέθη αυτά διαφέρουν μεταξύ τους και προσαρμόστηκαν κατάλληλα για τις ανάγκες της κάθε μετρικής της οποίας τα στατιστικά δεδομένα θέλουμε να εξάγουμε. Παρακάτω παρουσιάζεται μια λίστα με τα δεδομένα που λάβαμε από το JASP.

Μετρική 1 – Μέρες για την αναβάθμιση των πακέτων

Πίνακας 3-Μετρική 1 Αποτελέσματα

Project name	Min	Median	Max	Project name	Min	Median	Max
Amazeui	0 years and 1 days	1 years and 141 days	3 years and 256 days	**Bootstrap**	0 years and 0 days	1 years and 35 days	5 years and 249 days
Anime	0 years and 260 days	3 years and 232 days	3 years and 350 days	**browserify**	0 years and 1 days	4 years and 110 days	9 years and 163 days
Appwrite	0 years and 10 days	3 years and 35 days	6 years and 73 days	**Card**	0 years and 0 days	1 years and 212 days	7 years and 8 days
Async	0 years and 250 days	3 years and 210 days	4 years and 137 days	**Cesium**	0 years and 1 days	3 years and 245 days	9 years and 54 days
Ava	0 years and 0 days	3 years and 200 days	5 years and 223 days	**ChakraCore**	1 years and 165 days	4 years and 130 days	6 years and 182 days
Awx	0 years and 10 days	0 years and 90 days	4 years and 92 days	**Chart.js**	0 years and 4 days	0 years and 256 days	5 years and 250 days
Axios	0 years and 100 days	1 years and 290 days	6 years and 89 days	**clipboard.js**	0 years and 0 days	1 years and 188 days	7 years and 265 days
Backbone Marionette	0 years and 0 days	1 years and 236 days	4 years and 206 days	**Cropper.js**	0 years and 0 days	1 years and 140 days	7 years and 119 days

Project name	Min	Median	Max	Project name	Min	Median	Max
Cytoscape.js	0 years and 127 days	1 years and 266 days	7 years and 203 days	**Intro.js**	0 years and 1 days	2 years and 95 days	6 years and 147 days
Dash.js	0 years and 16 days	3 years and 85 days	7 years and 223 days	**Jasmine**	0 years and 13 days	3 years and 89 days	5 years and 264 days
Discord.js	0 years and 0 days	1 years and 85 days	4 years and 187 days	**Johnny-five**	0 years and 0 days	2 years and 230 days	7 years and 296 days
Dropzone	0 years and 4 days	1 years and 199 days	5 years and 310 days	**jsPDF**	0 years and 3 days	1 years and 280 days	8 years and 192 days
EaselJS	0 years and 65 days	2 years and 305 days	4 years and 209 days	**Karma**	0 years and 3 days	2 years and 119 days	8 years and 218 days
Editor.md	0 years and 4 days	1 years and 237 days	4 years and 209 days	**Knex**	0 years and 0 days	1 years and 355 days	5 years and 172 days
Element	0 years and 11 days	3 years and 115 days	3 years and 293 days	**Lazysizes**	0 years and 23 days	5 years and 25 days	6 years and 246 days
Eslint	0 years and 0 days	2 years and 355 days	8 years and 262 days	**Leetcode**	0 years and 58 days	2 years and 165 days	2 years and 186 days
Express	0 years and 0 days	3 years and 229 days	4 years and 163 days	**Magic Mirror**	0 years and 6 days	1 years and 343 Days	7 years and 320 days
Fetch	0 years and 6 days	3 years and 100 days	6 years and 267 days	**Material**	0 years and 6 days	1 years and 159 days	6 years and 218 days
Fine-uploader	0 years and 5 days	1 years and 140 days	4 years and 322 days	**Medium-Editor**	0 years and 14 days	1 years and 222 days	2 years and 197 days
Fingerprint.js	0 years and 7 days	1 years and 212 days	4 years and 341 days	**Mocha**	0 years and 2 days	1 years and 160 days	5 years and 194 days
Grunt	0 years and 0 days	3 years and 210 days	9 years and 65 days	**Modernizr**	0 years and 1 days	1 years and 227 days	7 years and 110 days
Hands Ontable	0 years and 0 days	1 years and 173 days	8 years and 36 days	**MQTT.js**	0 years and 0 days	2 years and 175 days	7 years and 308 days
Hexo	0 years and 0 days	1 years and 252 days	8 years and 226 days	**Mustache.js**	0 years and 13 days	2 years and 0 days	4 years and 38 days
Highlight.js	0 years and 1 days	2 years and 170 days	7 years and 266 days	**Node-soap**	0 years and 4 days	2 years and 124 days	8 years and 232 days
Images Loaded	0 years and 2 days	0 years and 90 days	1 years and 278 days	**NoVNC**	1 years and 13 days	2 years and 176 days	5 years and 90 days

Project name	Min	Median	Max	Project name	Min	Median	Max
Openlayers	0 years and 0 days	2 years and 40 days	7 years and 216 days	**Svgedit**	0 years and 0 days	1 years and 40 days	3 years and 264 days
p5.js	0 years and 48 days	2 years and 110 days	6 years and 284 days	**Svgo**	0 years and 1 days	1 years and 110 days	4 years and 28 days
Paper.js	0 years and 0 days	2 years and 130 days	7 years and 70 days	**Sweetalert2**	0 years and 1 days	2 years and 75 days	5 years and 181 days
Pdf.js	0 years and 1 days	1 years and 131 days	6 years and 155 days	**Typed.js**	0 years and 21 days	3 years and 55 days	6 years and 183 days
Pdfmake	0 years and 0 days	1 years and 131 days	4 years and 267 days	**Typeorm**	0 years and 26 days	1 years and 163 days	5 years and 53 days
Perfect-Scrollbar	0 years and 9 days	1 years and 265 days	3 years and 168 days	**Uglify.js**	0 years and 5 days	2 years and 108 days	2 years and 266 days
Pouchdb	0 years and 0 days	1 years and 195 days	5 years and 133 days	**Validator.js**	0 years and 42 days	2 years and 241 days	4 years and 133 days
Progress Bar.js	0 years and 3 days	1 years and 196 days	3 years and 223 days	**Velocity**	0 years and 4 days	1 years and 135 days	2 years and 258 days
Ramda	0 years and 4 days	1 years and 159 days	6 years and 93 days	**Video.js**	0 years and 0 days	2 years and 154 days	9 years and 124 days
ScrollMagic	0 years and 21 days	1 years and 253 days	5 years and 25 days	**Webogram**	0 years and 1 days	2 years and 200 days	3 years and 305 days
Sequelize	0 years and 17 days	1 years and 231 days	6 years and 198 days	**Webtorrent**	0 years and 0 days	1 years and 108 days	6 years and 180 days
Showdown	0 years and 6 days	2 years and 15 days	5 years and 191 days	**Ws**	0 years and 9 days	3 years and 130 days	5 years and 146 days
Sockjs-client	0 years and 2 days	2 years and 335 days	7 years and 161 days	**Wtfjs**	0 years and 12 days	4 years and 240 days	6 years and 197 days
Stackedit	0 years and 2 days	1 years and 350 days	5 years and 143 days	**x-spread sheet**	0 years and 6 days	2 years and 126 days	3 years and 86 days
Summer Note	0 years and 1 days	1 years and 208 days	8 years and 80 days	**Zero clipboard**	0 years and 30 days	1 years and 5 days	3 years and 237 days

Πίνακας 4-Βασικά Μεγέθη Μετρικής 1

	Minimum	Median	Maximum
Valid	80	80	80
Missing	0	0	0
Median	0.004	1.353	5.287
Std. Deviation	0.174	0.984	1.874
Minimum	0.000	0.040	1.278
Maximum	1.165	5.025	9.163

Valid: Ο αριθμός των έγκυρων δεδομένων που συμμετείχαν στην ανάλυση

Missing: Ο αριθμός των δεδομένων που απουσιάζουν από την ανάλυση

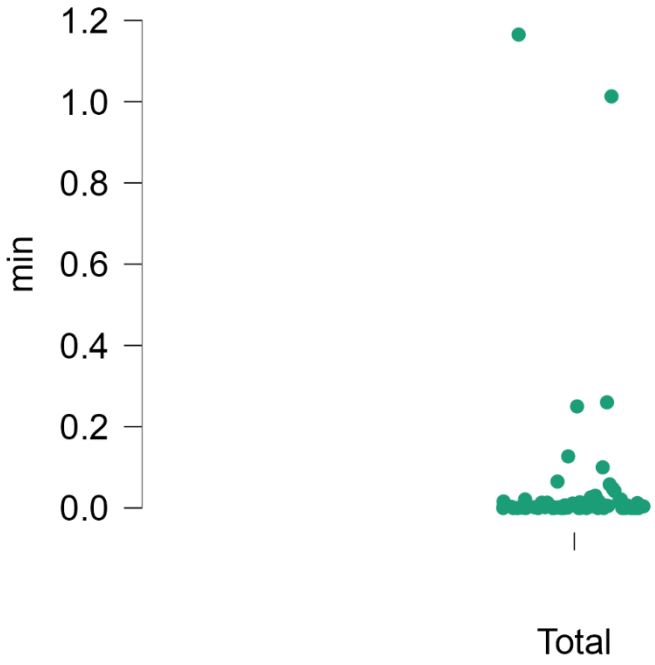
Median: Το άθροισμα όλων των δεδομένων διαιρείται με το δύο

Std.Deviation: Το ποσό της διασποράς του συνόλου των δεδομένων

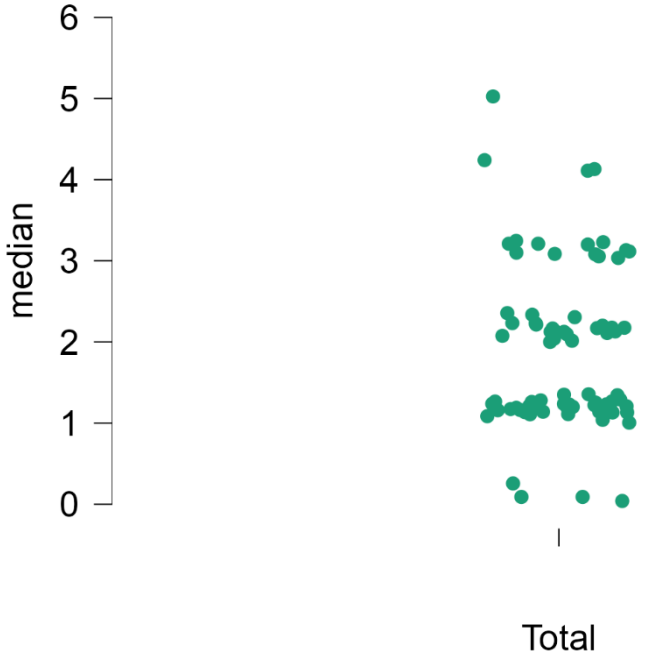
Minimum: Ελάχιστη τιμή των δεδομένων

Maximum: Μέγιστη τιμή των δεδομένων

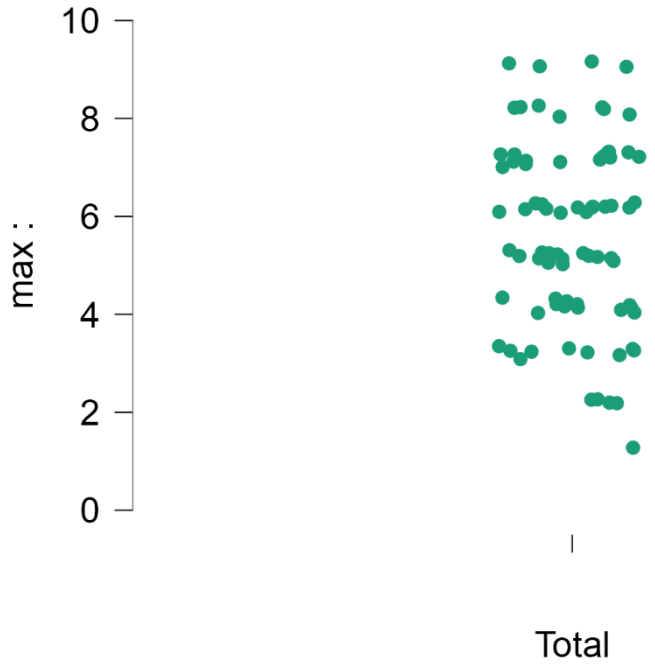
Σε αυτό το σημείο θα πρέπει να εξηγήσουμε ότι τα λογισμικά στατιστικής ανάλυσης δεν έχουν την δυνατότητα να αναγνωρίσουν το κείμενο ως ημερομηνία έτσι έγινε η κατάλληλη τροποποίηση ώστε να μπορέσουμε να εξάγουμε τα αποτελέσματα μας. Το αριστερό δεκαδικό ψηφίο υποδηλώνει τον χρόνο και τα δεξιά δεκαδικά ψηφία υποδηλώνουν τις μέρες π.χ. το 9.163 μεταφράζεται σε 9 χρόνια και 163 ημέρες



Σχήμα 3-Minimum Boxplot Μετρικής 1

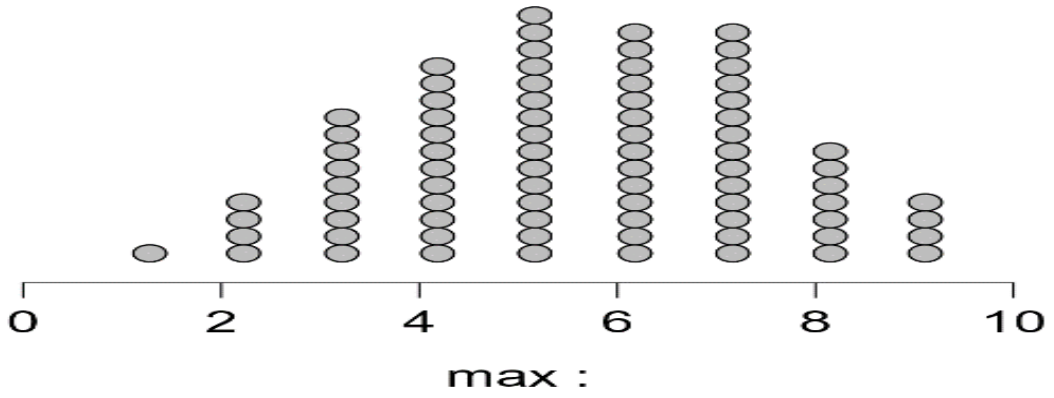


Σχήμα 4-Median Boxplot Μετρικής 1

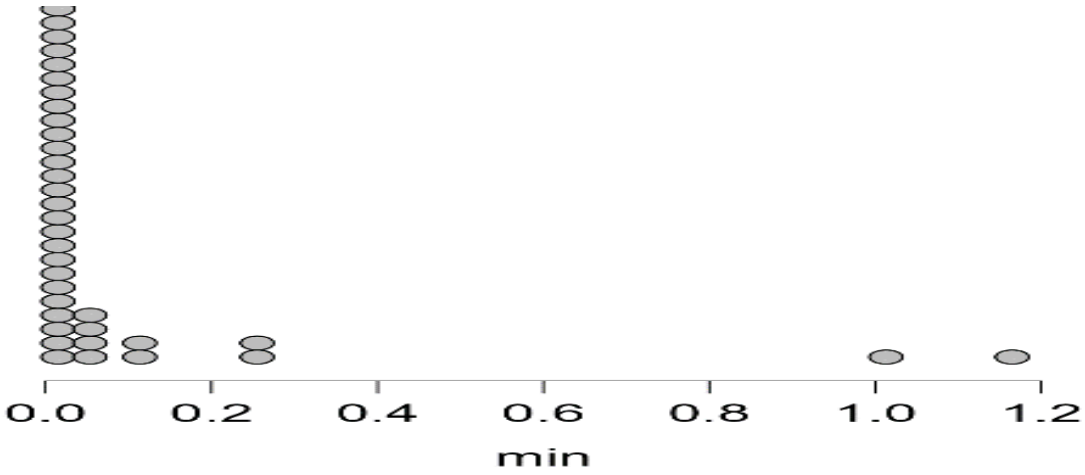


Σχήμα 5-Maximum Boxplot Μετρικής 1

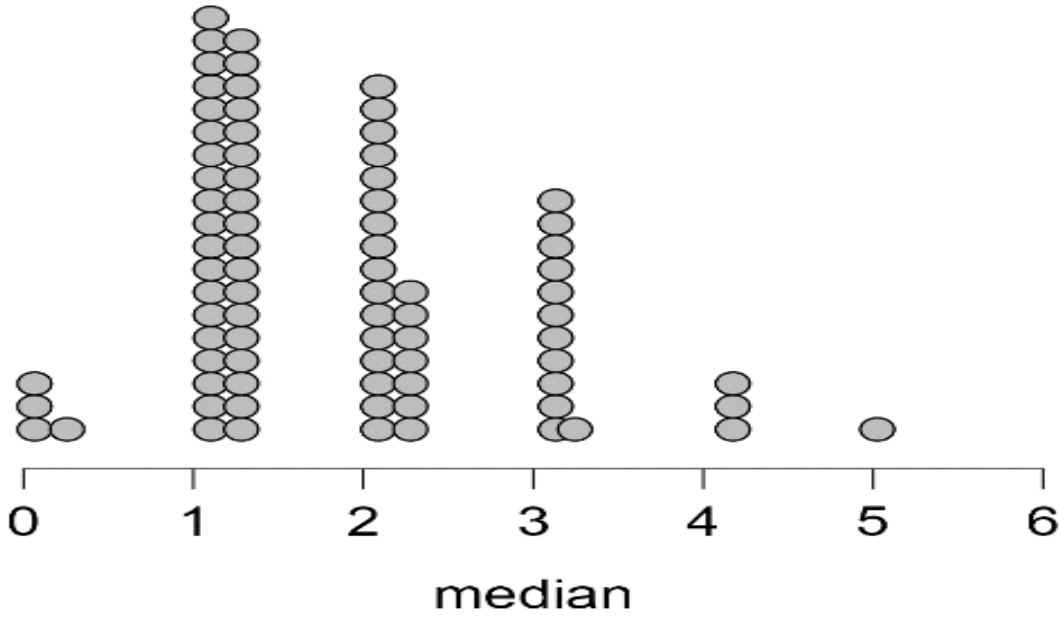
Τα παραπάνω αποτελέσματα δημιουργήθηκαν με την επιλογή boxplots>jigger form και αποτυπώνουν την συνολική κατανομή των δεδομένων πάνω στο καρτεσιανό σύστημα σε μορφή τελείας.



Σχήμα 6-Maximum Dot plot Μετρικής 1



Σχήμα 7-Minimum Dot plot Μετρικής 1



Σχήμα 8-Median Dot plot Μετρικής 1

Στα σχήματα 7,8,9 αποτυπώνονται τα αποτελέσματα κατανεμημένα μόνο στον οριζόντιο άξονα σε μορφή τελείας .

Μετρική 2 - Συνολικός αριθμός των μοναδικών πακέτων

Πίνακας 5-Μετρική 2 Αποτελέσματα

Project Name	Min	Std. Deviation	Max	Project name	Min	Std. Deviation	Max
Amazeui	31	3.270	40	**Editor.md**	13	1.304	16
Anime	1	2.138	5	**Element**	79	0.484	80
Appwrite	5	2.556	11	**Eslint**	85	1.661	90
Async	37	0.983	40	**Express**	28	6.510	55
Ava	74	7.929	99	**Fetch**	6	5.475	17
Awx	44	2.875	51	**Fine-uploader**	11	0.602	14
Axios	34	1.263	38	**Finger print.js**	5	14.696	39
Backbone marionette	30	5.035	48	**Grunt**	23	0.447	24
Bootstrap	39	4.640	59	**Hands ontable**	47	48.729	181
browserify	61	0.877	64	**Hexo**	34	2.114	40
Card	20	1.620	25	**Highlight. js**	17	4.479	35
Cesium	47	10.972	81	**Images loaded**	15	0.000	15
ChakraCore	5	2.046	15	**Intro.js**	1	16.536	41
Chart.js	53	2.999	28	**Jasmine**	11	4.092	22
clipboard.js	17	2.920	28	**Johnny-five**	22	0.702	24
Cropper.js	20	6.217	43	**jsPDF**	2	18.938	56
Cytoscape.js	30	0.000	30	**Karma**	63	5.734	77
Dash.js	43	3.428	55	**Knex**	48	1.850	56
Discord.js	11	7.118	29	**Lazysizes**	15	1.605	19
Dropzone	11	7.017	27	**Leetcode**	1	0.447	2
EaselJS	7	2.950	14	**Magic Mirror**	19	8.649	45

Project Name	Min	Std. Deviation	Max	Project name	Min	Std. Deviation	Max
Material	37	7.854	67	**Show down**	18	0.794	20
Medium-Editor	22	1.032	25	**Sockjs-client**	23	2.324	28
Mocha	81	11.310	109	**Stackedit**	84	6.771	101
Modernizr	27	1.797	33	**Summer Note**	13	17.734	65
MQTT.js	35	1.067	39	**Svgedit**	29	7.312	65
Mustache.js	2	2.100	10	**Svgo**	18	2.945	27
Node-soap	16	6.580	41	**Sweet alert2**	29	0.528	31
NoVNC	20	9.438	43	**Typed.js**	3	8.961	25
Openlayers	45	9.356	68	**Typeorm**	72	0.548	73
p5.js	30	11.814	64	**Uglify.js**	2	0.186	3
Paper.js	32	2.000	39	**Validator. Js**	13	1.069	16
Pdf.js	7	23.038	67	**Velocity**	3	6.389	20
Pdfmake	21	3.717	36	**Video.js**	70	2.573	77
Perfect-Scrollbar	5	4.834	21	**Web ogram**	17	4.503	32
Pouchdb	19	7.548	39	**Web torrent**	60	0.528	62
Progress Bar.js	6	6.667	29	**Ws**	12	0.986	13
Ramda	19	7.548	39	**Wtfjs**	12	0.447	13
ScrollMagic	10	8.167	27	**x-spread sheet**	25	0.000	25
Sequelize	53	2.684	63	**Zero clipboard**	7	5.247	22

Πίνακας 6-Βασικά Μεγέθη Μετρικής 2

	Minimum	Std.Deviation	Maximum
Valid	80	80	80
Missing	0	0	0
Median	20.000	2.974	37.000
Std. Deviation	22.131	6.825	28.715
Skewness	1.070	3.769	1.772
Std.Error of Skewness	0.269	0.269	0.269
Minimum	1.000	0.000	2.000
Maximum	85.000	48.729	181.000

Valid: Ο αριθμός των έγκυρων δεδομένων που συμμετείχαν στην ανάλυση

Missing: Ο αριθμός των δεδομένων που απουσιάζουν από την ανάλυση

Median: Το άθροισμα όλων των δεδομένων διαιρείται με το δύο

Std.Deviation: Το ποσό της διασποράς του συνόλου των δεδομένων

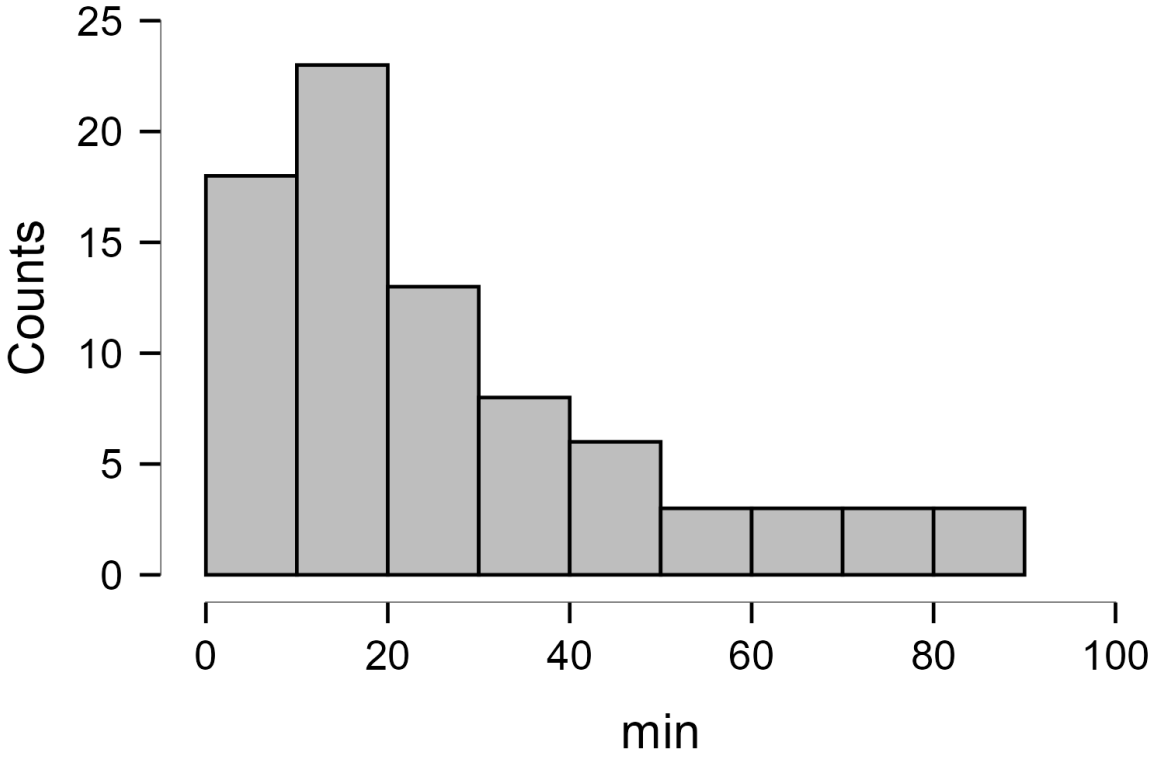
Skewness: Αριθμός που δείχνει πόσο ασύμμετρη είναι η κατανομή μας σε σχέση με μια κανονική κατανομή

Std Error of Skewness: Εάν αυτός ο αριθμός είναι μεγαλύτερος του +2 και μικρότερος του -2 αυτόματα καταλαβαίνουμε ότι η κατανομή μας δεν είναι κανονική

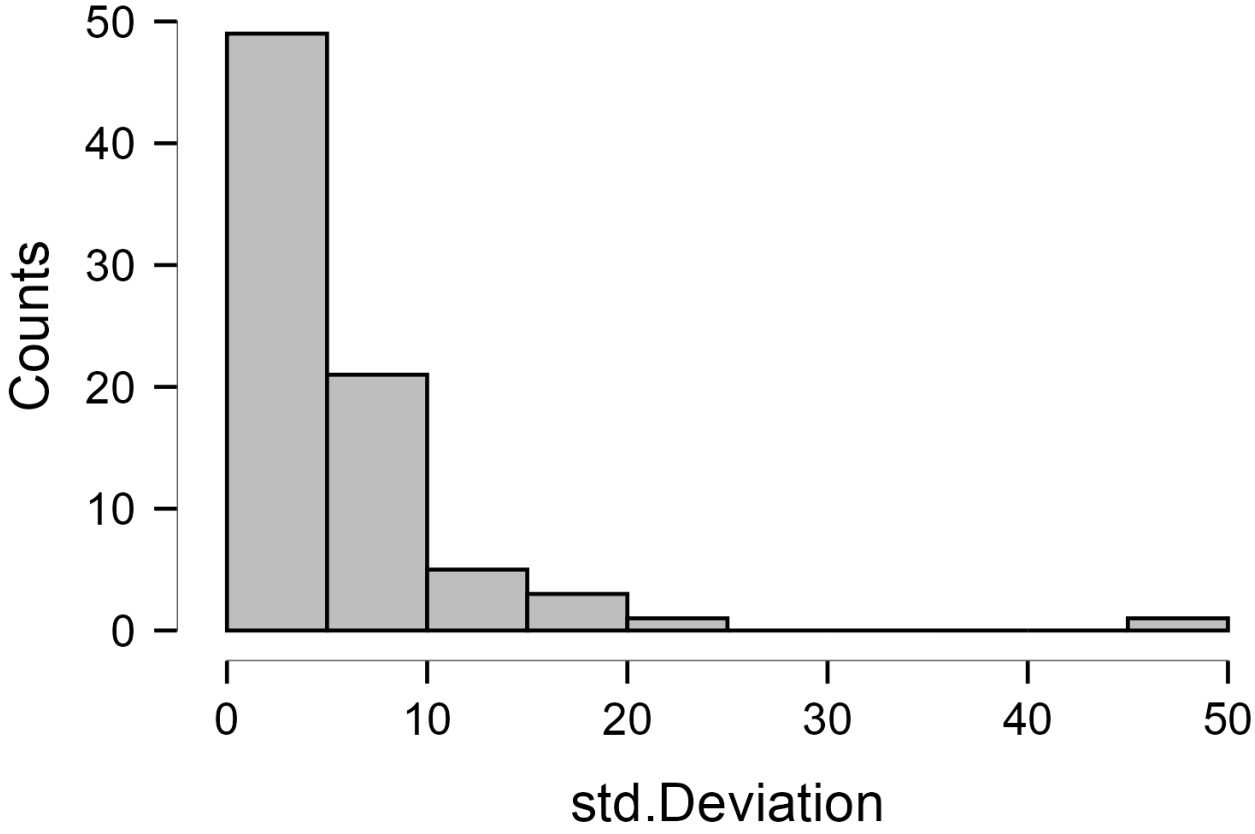
Minimum: Ελάχιστη τιμή των δεδομένων

Maximum: Μέγιστη τιμή των δεδομένων

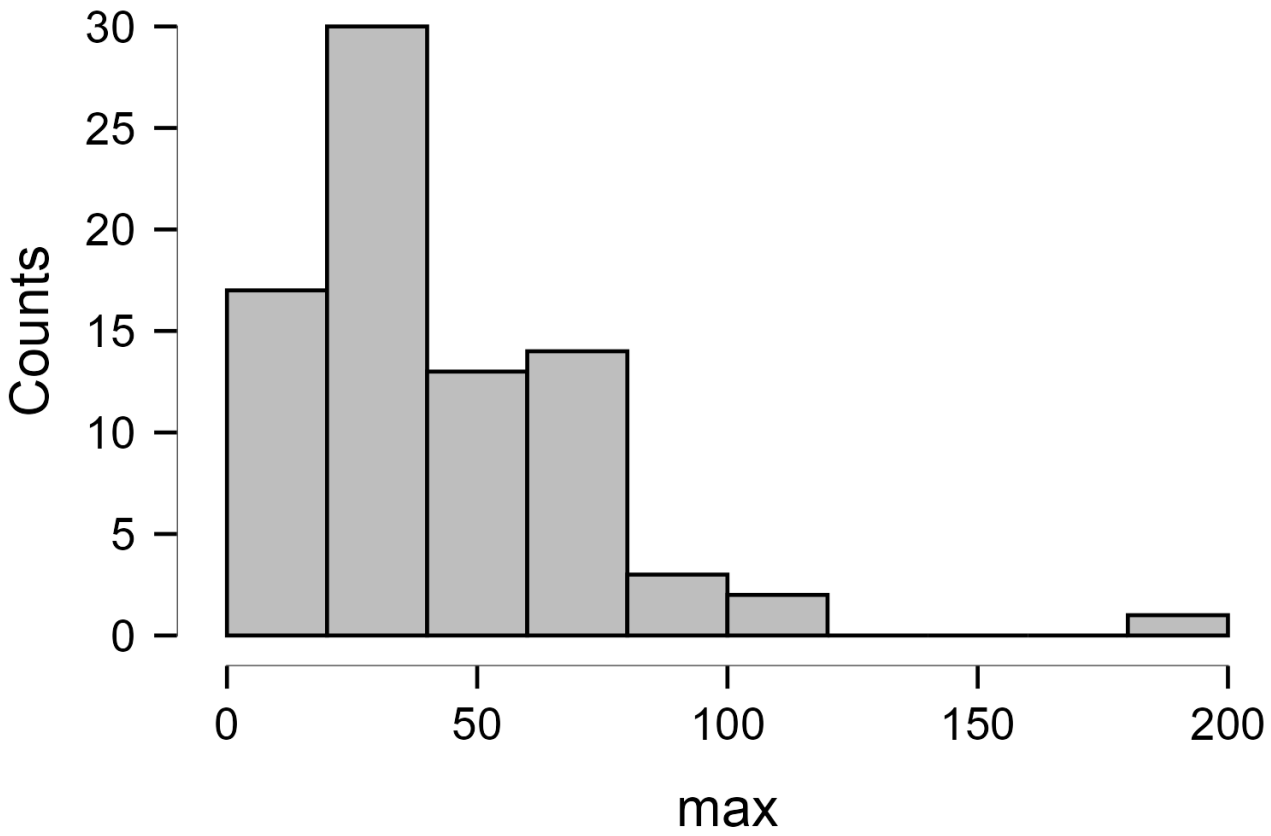
Σε αυτήν την μετρική πρέπει να δώσουμε ιδιαίτερη βαρύτητα στην στήλη Std.Deviation καθώς μας ενδιαφέρει να δούμε εάν τα πακέτα κάθε έκδοσης στα έργα μας ακολουθούν κανονική κατανομή (Συμμετρική ως προς τον μέσο όρο) για να δούμε την τάση που έχουν να γίνονται αλλαγές ή όχι με την πάροδο του χρόνου. Περισσότερες λεπτομέρειες για τα συμπεράσματα θα αναλυθούν στο επόμενο κεφάλαιο.



Σχήμα 9-Minimum Distribution Plot Μετρικής 2

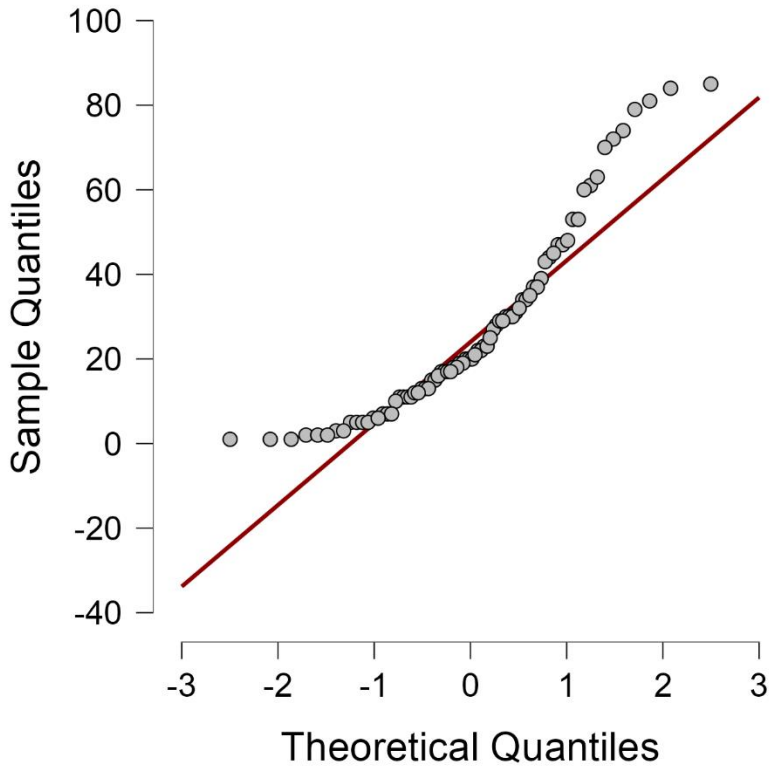


Σχήμα 10-Std.Deviation Distribution Plot Μετρικής 2

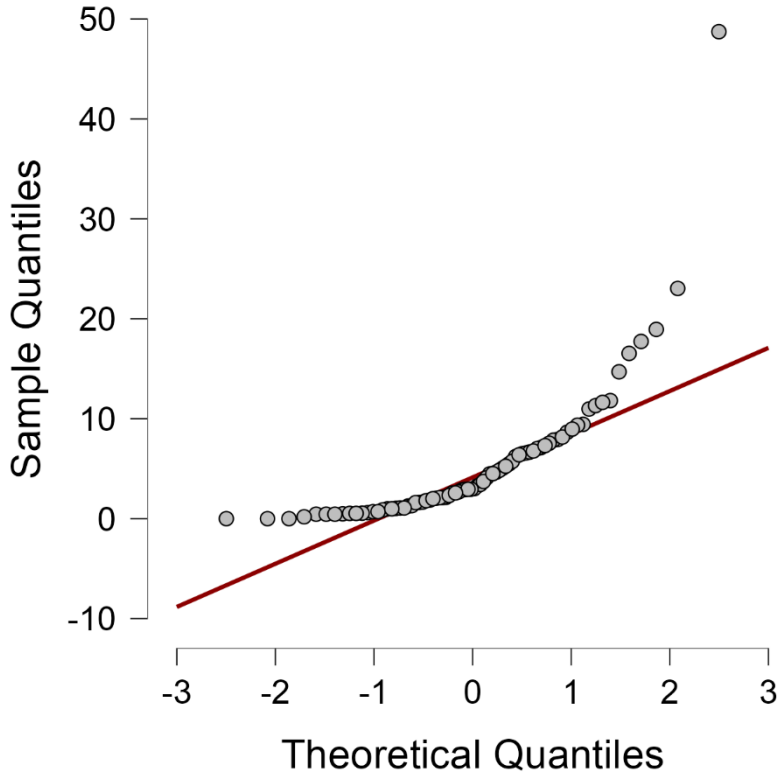


Σχήμα 11-Maximum Distribution Plot Μετρικής 2

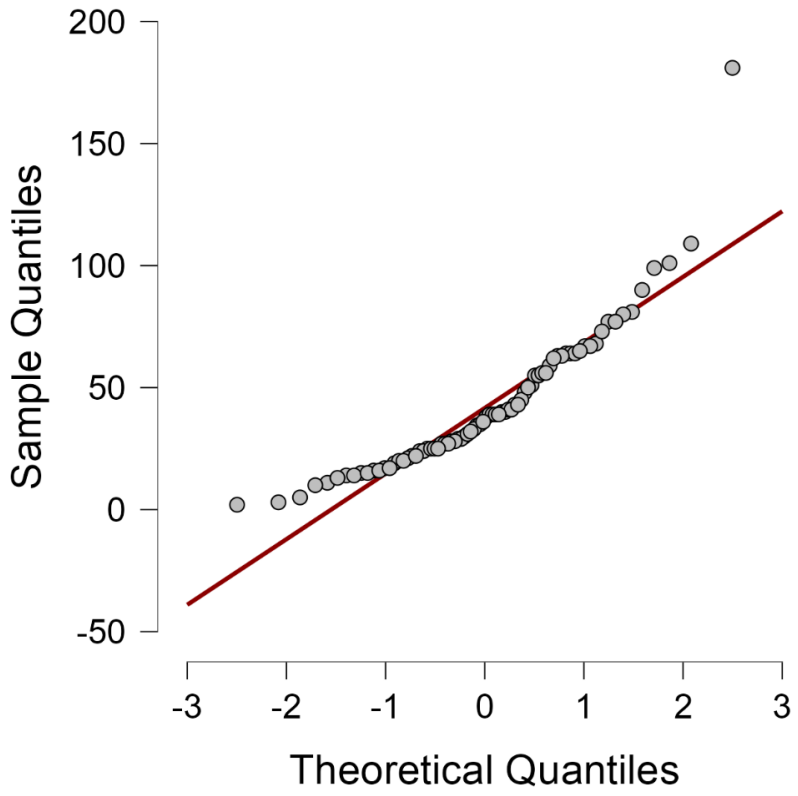
Τα Distribution plots χωρίζουν τα δεδομένα μας σε μικρά διαστήματα και υποδηλώνουν την συχνότητα που εμφανίζονται τα δεδομένα μας. Όσο πιο κοντά σε σχήμα «καμπάνας» είναι η κατανομή μας τόσο περισσότερο τείνει να θεωρείται κανονική κατανομή. Για αυτόν ακριβώς τον λόγο επιλέχθηκαν για να μας βοηθήσουν να αξιολογήσουμε κατά πόσο οι κατανομές μας είναι κανονικές ή όχι.



Σχήμα 12-Minimum Q-Q Plot Μετρικής 2



Σχήμα 13-Std.Deviation Q-Q Plot Μετρικής 2



Σχήμα 14-Maximum Q-Q Plot Μετρικής 2

Τα Q-Q plots χρησιμοποιούν μια γραμμή γωνίας 45 μοιρών για να μελετήσουν κατά πόσο η κατανομή μας είναι κανονική. Όσο περισσότερο τα σημεία μιας κατανομής εμφανίζονται κυρίως πάνω σε αυτήν την γραμμή τόσο περισσότερο συγκλίνει στην κανονικότητα. Επιλέχθηκαν για να μελετήσουμε την κανονικότητα της κατανομής μας

Μετρική 3 - Αριθμός καινούριων πακέτων που προστέθηκαν για κάθε project

Πίνακας 7-Μετρική 3 Αποτελέσματα

Project name	Min	Mean	Max	Project name	Min	Mean	Max
Amazeui	0	2	10	**Editor.md**	0	0.750	2
Anime	0	1	5	**Element**	0	0.083	2
Appwrite	0	0.462	2	**Eslint**	0	0.538	1
Async	0	0.8	3	**Express**	0	2	18
Ava	0	1.467	11	**Fetch**	0	1.059	12
Awx	0	0.8	3	**Fine-uploader**	0	0.111	1
Axios	0	1	4	**Finger print.js**	0	1.438	7
Backbone Marionette	0	1.273	10	**Grunt**	0	0	0
Bootstrap	0	4.7	14	**Hands ontable**	0	3.769	19
Browserify	0	10.5	61	**Hexo**	0	0.647	5
Card	0	1.417	10	**Highlight. js**	0	2	9
Cesium	0	2.833	22	**Images loaded**	0	1	2
ChakraCore	0	0	0	**Intro.js**	0	2.438	23
Chart.js	0	0.444	1	**Jasmine**	0	0.429	4
clipboard.js	0	1.467	7	**Johnny-five**	0	0.071	1
Cropper.js	0	1.867	14	**jsPDF**	0	5.111	25
Cytoscape.js	0	0.250	4	**Karma**	0	0.2	1
Dash.js	0	1.983	22	**Knex**	0	1.231	4
Discord.js	0	2.063	19	**Lazysizes**	0	0.6	4
Dropzone	0	2.077	21	**Leetcode**	0	0.250	1
EaselJS	0	1.750	6	**Magic Mirror**	0	3.059	10

Project name	Min	Mean	Max	Project name	Min	Mean	Max
Material	0	1.4	5	**Show down**	0	0.167	1
Medium-Editor	0	0.3	2	**Sockjs-client**	0	1.250	5
Mocha	0	4.063	16	**Stackedit**	0	0.5	2
Modernizr	0	1.714	12	**Summer Note**	0	1.133	8
MQTT.js	0	0.368	4	**Svgedit**	1	3.167	6
Mustache.js	0	0.6	3	**Svgo**	0	1.7	9
Node-soap	0	1.357	8	**Sweet alert2**	0	0.067	1
NoVNC	1	16.2	32	**Typed.js**	0	2.5	18
Openlayers	0	3.357	22	**Typeorm**	0	0	0
p5.js	0	1.824	13	**Uglify.js**	0	0	0
Paper.js	0	0.706	2	**Validator. Js**	0	0.250	1
Pdf.js	0	3.667	10	**Velocity**	0	0.875	6
Pdfmake	0	1.2	12	**Video.js**	0	0.538	5
Perfect-Scrollbar	0	1.143	5	**Web ogram**	0	2.8	10
Pouchdb	0	2.455	19	**Web torrent**	0	0.071	1
Progress Bar.js	0	2	7	**Ws**	0	0.071	1
Ramda	0	2.7	7	**Wtfjs**	0	0.167	1
ScrollMagic	0	7	20	**x-spread sheet**	0	0	0
Sequelize	0	1.692	10	**Zero clipboard**	0	1	13

Πίνακας 8-Βασικά Μεγέθη Μετρικής 3

	Minimum	Mean	Maximum
Valid	80	80	80
Missing	0	0	0
Median	0.025	1.712	8.750
Std. Deviation	0.157	2.336	9.390
Skewness	6.202	3.953	2.603
Std. Error of Skewness	0.269	0.269	0.269
Minimum	0.000	0.000	0.000
Maximum	1.000	16.200	61.000

Valid: Ο αριθμός των έγκυρων δεδομένων που συμμετείχαν στην ανάλυση

Missing: Ο αριθμός των δεδομένων που απουσιάζουν από την ανάλυση

Mean: Μέσος Όρος των δεδομένων

Std. Deviation: Το ποσό της διασποράς του συνόλου των δεδομένων

Skewness: Αριθμός που δείχνει πόσο ασύμμετρη είναι η κατανομή μας σε σχέση με μια κανονική κατανομή

Std Error of Skewness: Εάν αυτός ο αριθμός είναι μεγαλύτερος του +2 και μικρότερος του -2 αυτόματα καταλαβαίνουμε ότι η κατανομή μας δεν είναι κανονική

Minimum: Ελάχιστη τιμή των δεδομένων

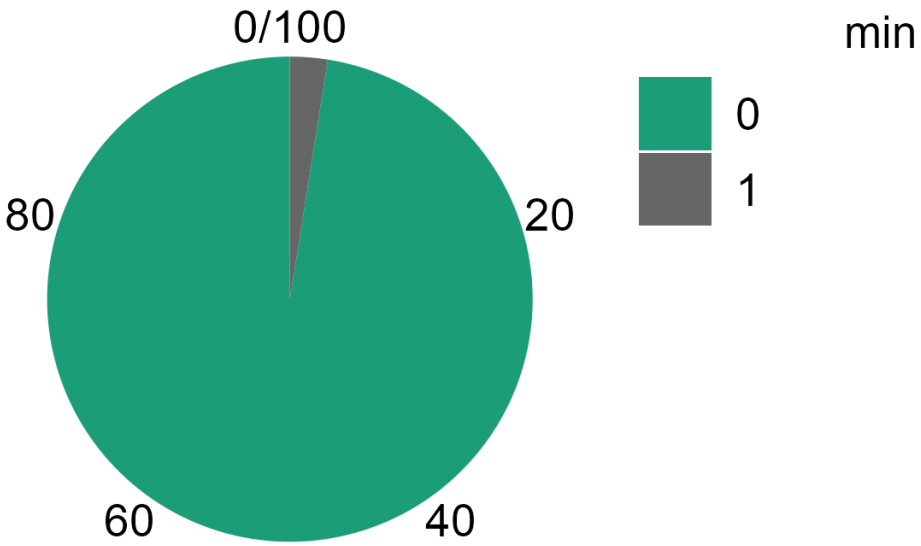
Maximum: Μέγιστη τιμή των δεδομένων

Πίνακας 9-Σημαντικά Αποτελέσματα Μετρικής 3

Min	Frequency	Percent
0	78	97.500
1	2	2.500
Missing	0	0.000
Total	80	100.000

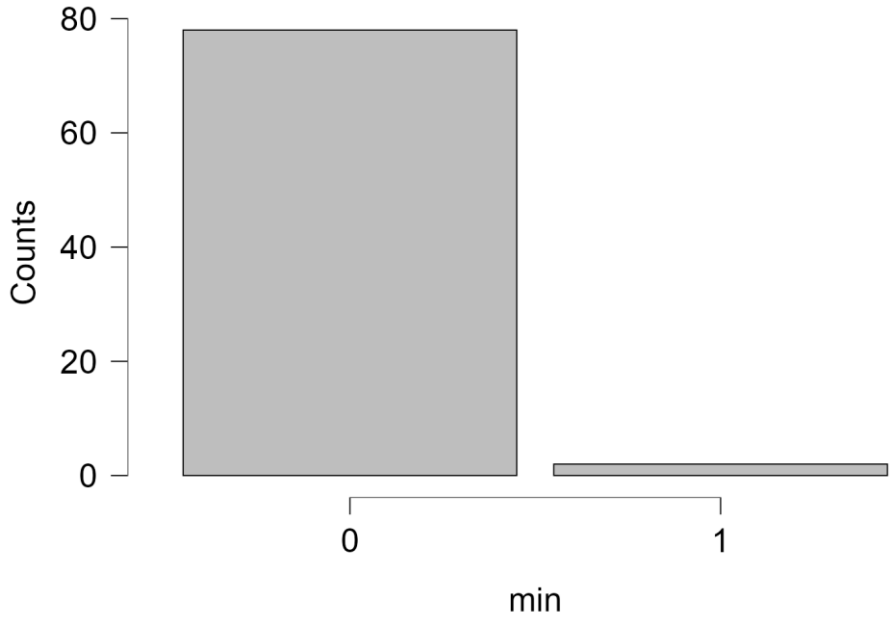
Frequency: Συχνότητα των δεδομένων

Percent: Ποσοστό επί τις 100

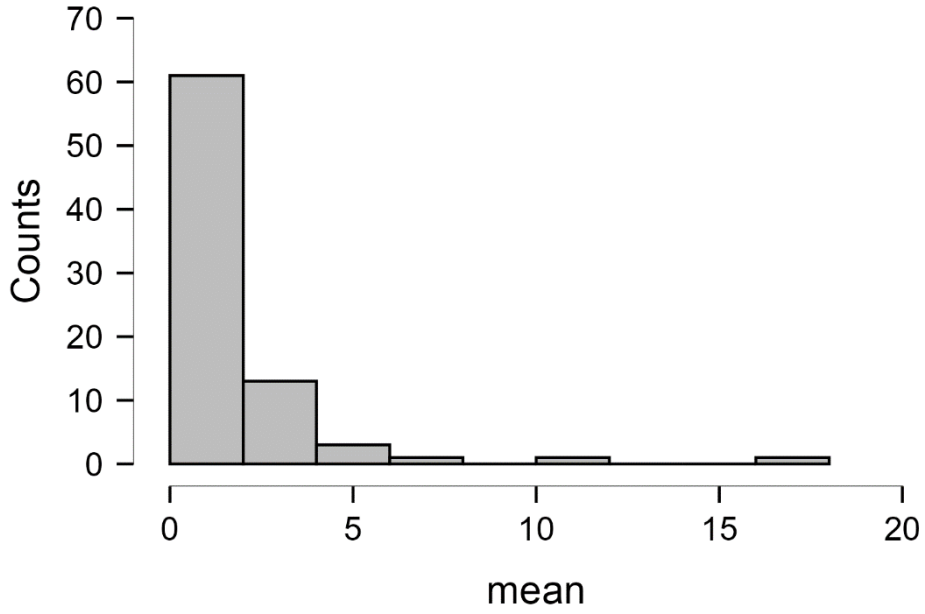


Σχήμα 15-Distribution Pie

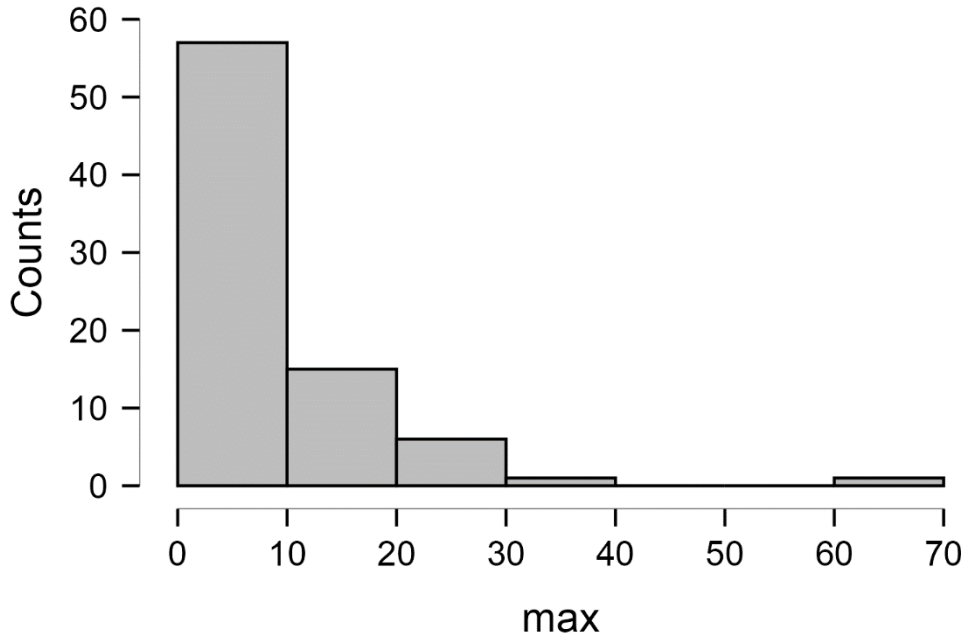
Τα παραπάνω αποτελέσματα παρουσιάζονται για να δείξουν ότι το 97.5 των έργων έχει έστω και ένα διαδοχικό ζευγάρι στο οποίο δεν προστέθηκε κανένα πακέτο.



Σχήμα 16-Minimum Distribution Plot Μετρικής 3



Σχήμα 17-Mean Distribution Plot Μετρικής 3



Σχήμα 18-Maximum Distribution Plot Μετρικής 3

Στην μετρική 3 παρατηρούμε εύκολα από τα Distribution plots ότι οι κατανομές μας δεν είναι κανονικές (Απουσία σχήματος «καμπάνας»)

Μετρική 4.1 - Διαίρεση με αριθμητή τον αριθμό των πακέτων που προστέθηκαν και αφαιρέθηκαν και παρονομαστή τον αριθμό των ημερών μεταξύ δύο διαδοχικών εκδόσεων

Πίνακας 10-Μετρική 4.1 Αποτελέσματα

Project name	Min	Mean	Max	Project name	Min	Mean	Max
amazeui	0	0.053	0.140	**Editor.md**	0	0.427	1.500
Anime	0	0	0	**Element**	0	0.105	1
appwrite	0	0.009	0.040	**Eslint**	0	0.045	0.280
async	0	0	0	**Express**	0	0.075	0.550
ava	0	0.139	1	**Fetch**	0	0.014	0.080
awx	0	0.040	0.130	**Fine-uploader**	0	0.010	0.090
axios	0	0.258	2	**Finger print.js**	0	0.051	0.410
Backbone marionette	0	0.053	0.270	**Grunt**	0	0	0
bootstrap	0	0.047	0.140	**Hands ontable**	0	0.049	0.330
browserify	0.010	0.025	0.040	**Hexo**	0	0.036	0.140
card	0	0.067	0.310	**Highlight. js**	0	0.051	0.230
cesium	0	0.110	0.700	**Images loaded**	0	0.003	0.010
ChakraCore	0	0	0	**Intro.js**	0	0.059	0.450
Chart.js	0	0.281	2.330	**Jasmine**	0	0.008	0.040
clipboard.js	0	0.417	4	**Johnny-five**	0	0.0009	0.010
Cropper.js	0	0.043	0.280	**jsPDF**	0	0.074	0.250
Cytoscape.js	0	0.010	0.130	**Karma**	0	0.079	0.500
Dash.js	0	0.062	0.720	**Knex**	0	0.085	0.500
Discord.js	0	0.071	0.680	**Lazysizes**	0	0.044	0.230
Dropzone	0	0.345	3.220	**Leetcode**	0	0.022	0.090
EaselJS	0	0.033	0.070	**Magic Mirror**	0	0.046	0.200

Project Name	Min	Mean	Max	Project name	Min	Mean	Max
Material	0	0.050	0.200	**Show down**	0	0.007	0.040
Medium-Editor	0	0.021	0.200	**Sockjs-client**	0	0.020	0.060
Mocha	0	0.157	0.420	**Stackedit**	0	0.010	0.020
Modernizr	0	0.017	0.100	**Summer Note**	0	0.426	5
MQTT.js	0	0.028	0.270	**Svgedit**	0.040	0.192	0.500
Mustache.js	0	0.002	0.010	**Svgo**	0	0.097	0.500
Node-soap	0	0.011	0.060	**Sweet alert2**	0	0.004	0.040
NoVNC	0.950	0.950	0.950	**Typed.js**	0	0.042	0.210
Openlayers	0	0.141	0.500	**Typeorm**	0	0	0
p5.js	0	0.025	0.100	**Uglify.js**	0	0.0006	0.010
Paper.js	0	0.617	3	**Validator. Js**	0	0.005	0.020
Pdf.js	0	0.278	2.500	**Velocity**	0	0.004	0.030
Pdfmake	0	0.018	0.120	**Video.js**	0	0.160	1
Perfect-Scrollbar	0	0.029	0.140	**Web ogram**	0	0.028	0.060
Pouchdb	0	0.093	0.330	**Web torrent**	0	0.193	1
Progress Bar.js	0	0.020	0.050	**Ws**	0	0.006	0.060
Ramda	0	0.207	1	**Wtfjs**	0	0	0
ScrollMagic	0	0.087	0.250	**x-spread sheet**	0	0	0
Sequelize	0	0.017	0.040	**Zero clipboard**	0	0.008	0.060

Πίνακας 11-Βασικά Μεγέθη Μετρικής 4.1

	Minimum	**Mean**	**Maximum**
Valid	80	80	79
Missing	0	0	1
Mean	0.013	0.092	0.532
Std. Deviation	0.106	0.152	0.925
Minimum	0.000	0.000	0.000
Maximum	0.950	0.950	5.000

Valid: Ο αριθμός των έγκυρων δεδομένων που συμμετείχαν στην ανάλυση

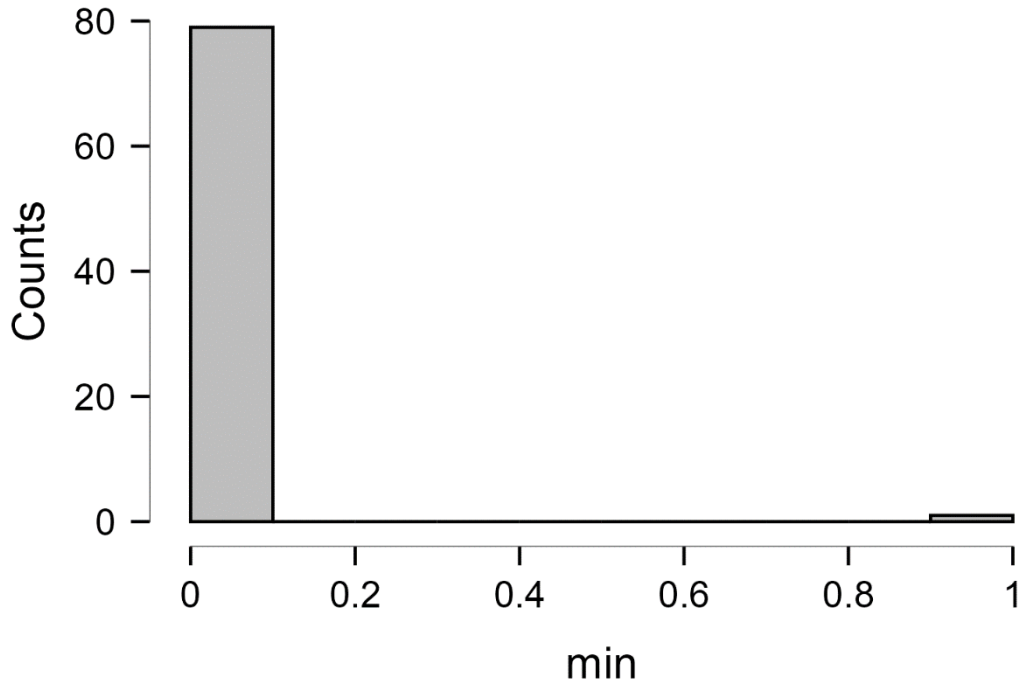
Missing: Ο αριθμός των δεδομένων που απουσιάζουν από την ανάλυση

Mean: Μέσος Όρος των δεδομένων

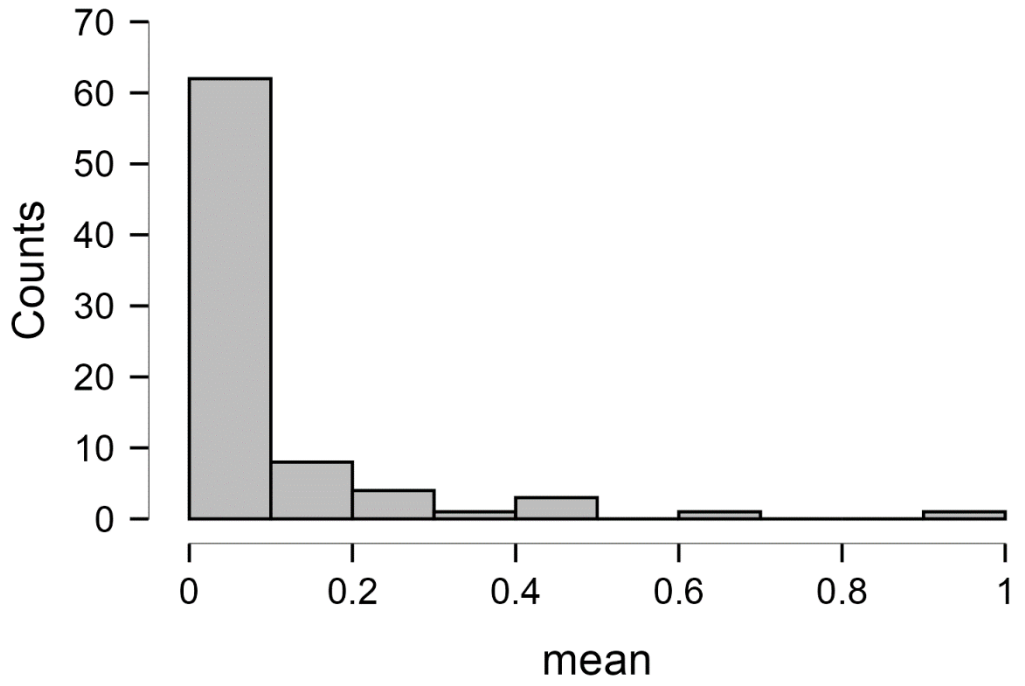
Std. Deviation: Το ποσό της διασποράς του συνόλου των δεδομένων

Minimum: Ελάχιστη τιμή των δεδομένων

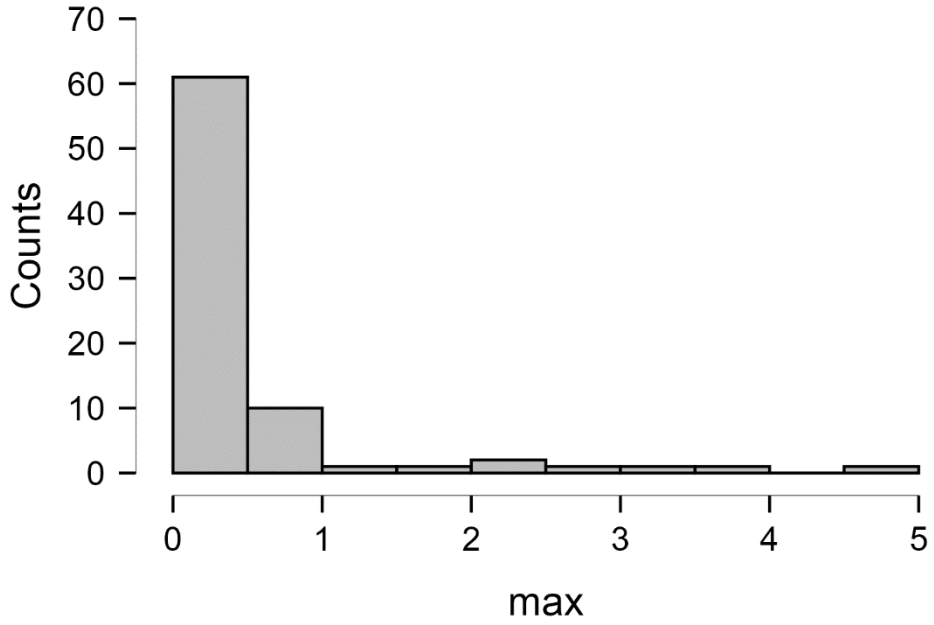
Maximum: Μέγιστη τιμή των δεδομένων



Σχήμα 19-Minimum Distribution Plot Μετρικής 4.1



Σχήμα 20-Mean Distribution Plot Μετρικής 4.1



Σχήμα 21-Maximum Distribution Plot Μετρικής 4.1

Από τα Distribution plots μπορούμε να βγάλουμε δύο συμπεράσματα. Πρώτον καμία κατανομή δεν είναι κανονική και δεύτερον και εξίσου σημαντικό βλέπουμε ότι και στις τρείς κατανομές μας το μεγαλύτερο ποσοστό κατανέμεται κοντά στο μηδέν. Το αποτέλεσμα αυτό ήταν σχεδόν αναμενόμενο αφού βλέπουμε αποτελέσματα διαίρεσης στην οποία είναι λογικό οι συνολικές αλλαγές στο πακέτο να είναι μικρότερος αριθμός από τις μέρες που χρειάστηκαν για να γίνει αναβάθμιση σε μια έκδοση.

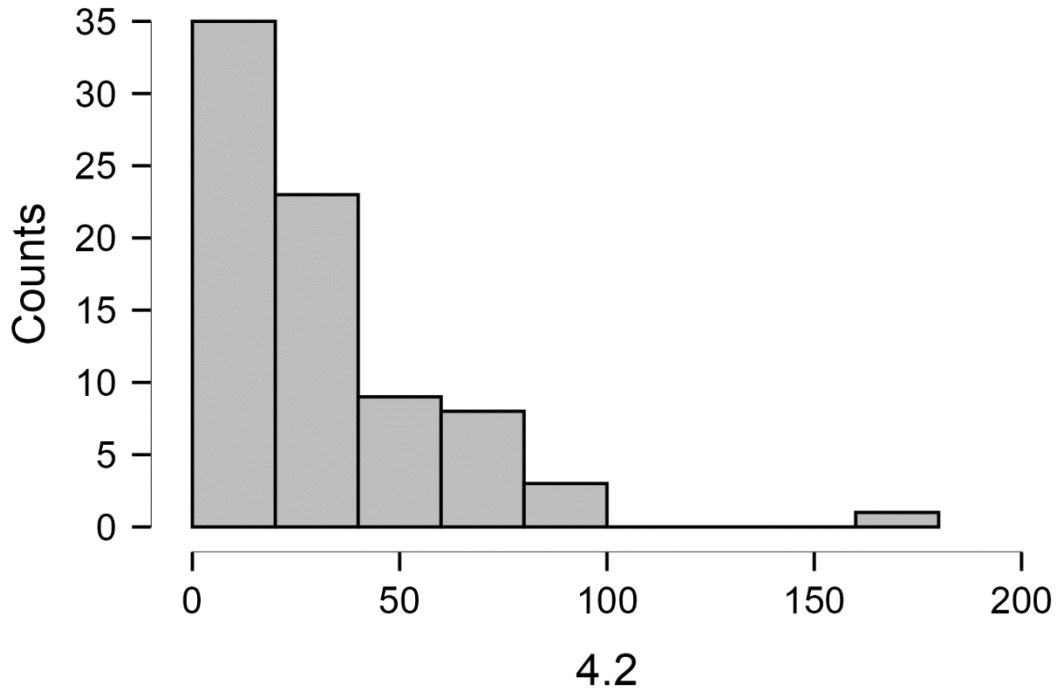
Μετρική 4.2 και 4.3 – Αριθμός Μοναδικών πακέτων στην τωρινή έκδοση και σε προηγούμενες εκδόσεις

Όπως αναφέραμε και στο κεφάλαιο 3 αυτές οι μετρικές θα εξεταστούν ταυτόχρονα, με σκοπό να συγκρίνουμε τα αποτελέσματα που θα προκύψουν.

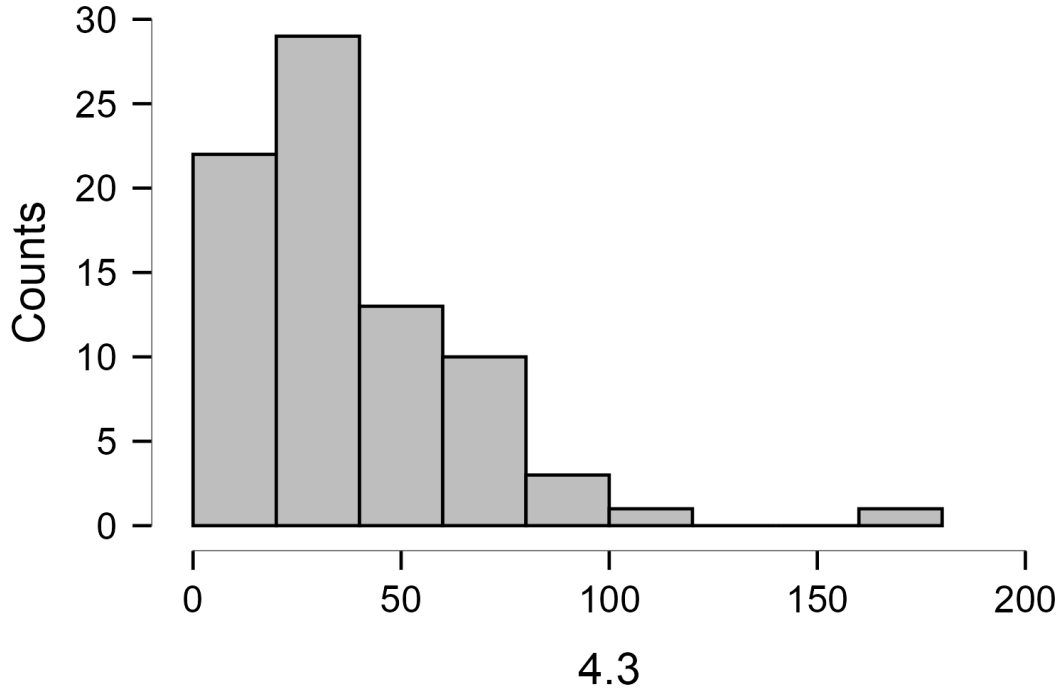
Πίνακας 12-Βασικά Μεγέθη Μετρικής 4.2 και 4.3

	First Versions
Valid	80
Missing	0
Mean	31.532
Std. Deviation	29.285
Minimum	1.000
Maximum	180.000

	Current Versions
Valid	80
Missing	0
Mean	38.658
Std. Deviation	27.994
Minimum	2.000
Maximum	179.000



Σχήμα 22-Distribution Plot 4.2



Σχήμα 23-Distribution Plot 4.3

Από τα βασικά μεγέθη και από τα Distribution Plots συμπεραίνουμε τελικά ότι υπάρχουν διαφορές αλλά δεν είναι μεγάλες οι διαφορές. Ο μέσος όρος των δύο αποτελεσμάτων διαφέρει για 7 μονάδες και όλα τα υπόλοιπα μεγέθη σχεδόν ταυτίζονται. Στα Distribution plots βλέπουμε ότι το μεγαλύτερο ποσοστό των πακέτων της τωρινής έκδοσης κατανέμονται δεξιότερα σε σχέση με τα πακέτα της πρώτης έκδοσης.

Μετρική 5 - Αριθμός των πακέτων που προστέθηκαν και αριθμός των πακέτων που αφαιρέθηκαν

Πίνακας 13-Μετρική 5 Αποτελέσματα

Project name	Min	Mean	Max	Project name	Min	Mean	Max
amazeui	0	4.636	13	**Editor.md**	0	2.250	7
anime	0	1.200	6	**Element**	0	0.467	4
appwrite	0	1.615	6	**Eslint**	0	1.077	7
async	0	1.000	4	**Express**	0	2.125	22
ava	0	4.286	40	**Fetch**	0	1.471	14
awx	0	1.200	6	**Fine-uploader**	0	0.222	2
axios	0	1.267	4	**Finger print.js**	0	1.625	7
Backbone marionette	0	4.273	30	**Grunt**	0	0.250	1
bootstrap	0	9.200	22	**Hands ontable**	0	4.538	19
browserify	0.010	21.00	122	**Hexo**	0	1.450	9
card	0	2.917	24	**Highlight. js**	0	2.200	9
cesium	0	3.250	22	**Images loaded**	0	2.000	4
ChakraCore	0	0	0	**Intro.js**	0	2.438	23
Chart.js	0	1.545	14	**Jasmine**	0	0.786	5
clipboard.js	0	2.533	11	**Johnny-five**	0	0.214	2
Cropper.js	0	2.467	15	**jsPDF**	0	7.222	35
Cytoscape.js	0	0.500	8	**Karma**	0	0.950	10
Dash.js	0	3.938	53	**Knex**	0	2.462	8
Discord.js	0	3.125	24	**Lazysizes**	0	1.000	4
Dropzone	0	2.929	29	**Leetcode**	0	0.250	1
EaselJS	0	2.500	7	**Magic Mirror**	0	5.765	19

Project name	Min	Mean	Max	Project name	Min	Mean	Max
Material	0	2.000	7	**Show down**	0	0.333	2
Medium-Editor	0	0.300	2	**Sockjs-client**	0	1.875	7
Mocha	0	6.125	21	**Stackedit**	1	0.500	2
Modernizr	0	3.714	26	**Summer Note**	0	2.333	10
MQTT.js	0	0.895	8	**Svgedit**	1	4.833	8
Mustache.js	0	0.600	3	**Svgo**	0	4.000	21
Node-soap	0	1.714	8	**Sweet alert2**	0	0.267	2
NoVNC	5	24.400	44	**Typed.js**	0	2.750	18
Openlayers	0	6.000	25	**Typeorm**	0	0	0
p5.js	0	3.882	21	**Uglify.js**	0	0.067	1
Paper.js	0	1.294	4	**Validator. Js**	0	0.500	1
Pdf.js	0	5.889	15	**Velocity**	0	0.875	6
Pdfmake	0	2.067	20	**Video.js**	0	0.615	5
Perfect-Scrollbar	0	2.786	19	**Web ogram**	0	3.200	10
Pouchdb	0	5.273	27	**Web torrent**	0	0.357	2
Progress Bar.js	0	3.667	19	**Ws**	0	0.143	1
Ramda	0	3.600	11	**Wtfjs**	0	0.167	1
ScrollMagic	0	9.750	24	**x-spread sheet**	0	0	0
Sequelize	0	2.846	17	**Zero clipboard**	0	1.500	16

Πίνακας 14-Βασικά Μεγέθη Μετρικής 5

	Minimum	Median	Maximum
Valid	80	80	80
Missing	0	0	0
Mean	0.075	2.841	13.787
Std. Deviation	0.569	3.815	16.455
Minimum	0.000	0.000	0.000
Maximum	5.000	24.400	122.000

Valid: Ο αριθμός των έγκυρων δεδομένων που συμμετείχαν στην ανάλυση

Missing: Ο αριθμός των δεδομένων που απουσιάζουν από την ανάλυση

Mean: Μέσος Όρος των δεδομένων

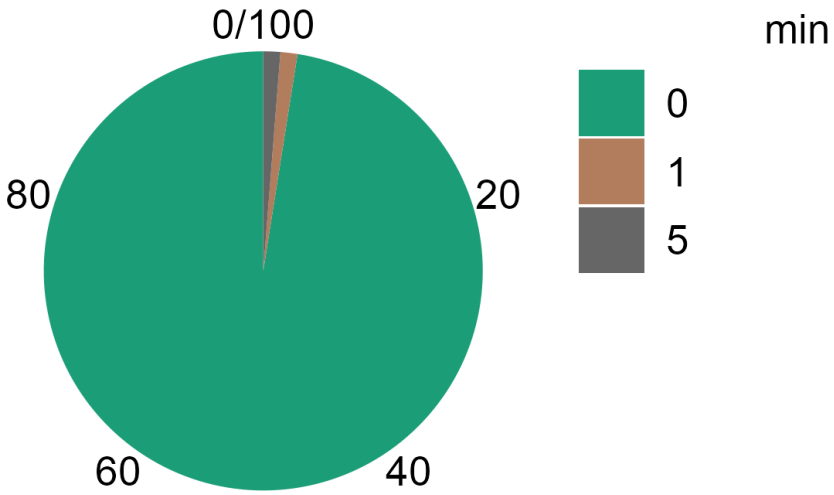
Std.Deviation: Το ποσό της διασποράς του συνόλου των δεδομένων

Minimum: Ελάχιστη τιμή των δεδομένων

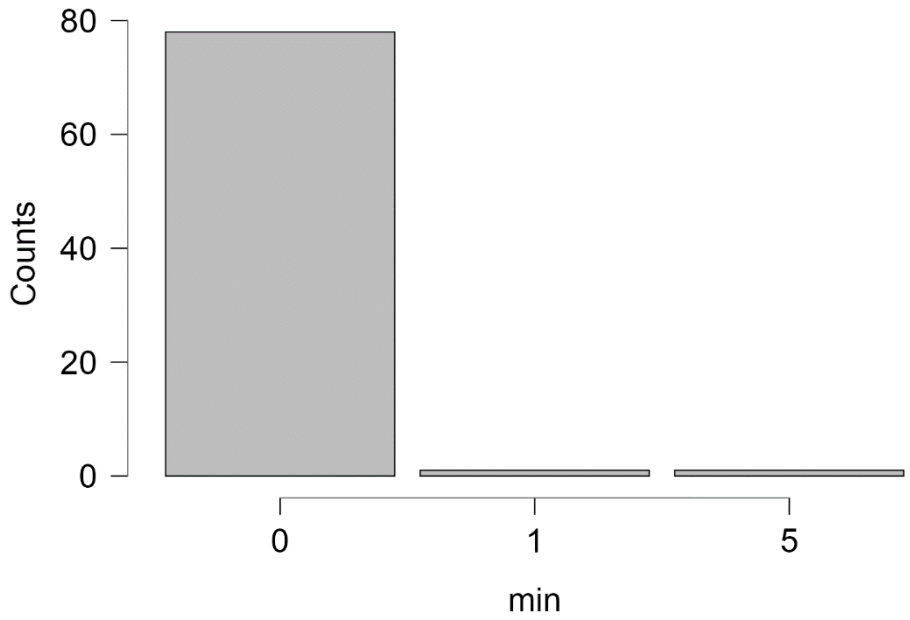
Maximum: Μέγιστη τιμή των δεδομένων

Πίνακας 15-Σημαντικά Αποτελέσματα Μετρικής 5

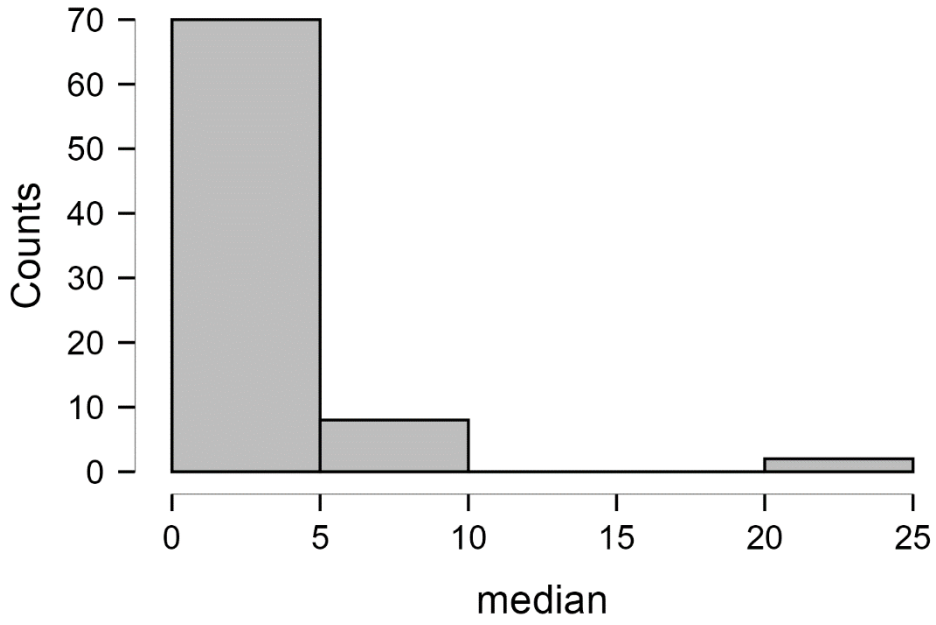
Min	Frequency	Percent	Valid Percent
0	78	97.500	97.500
1	1	1.250	1.250
5	1	1.250	1.250
Missing	0	0.000	0.000
Total	80	100.000	100.000



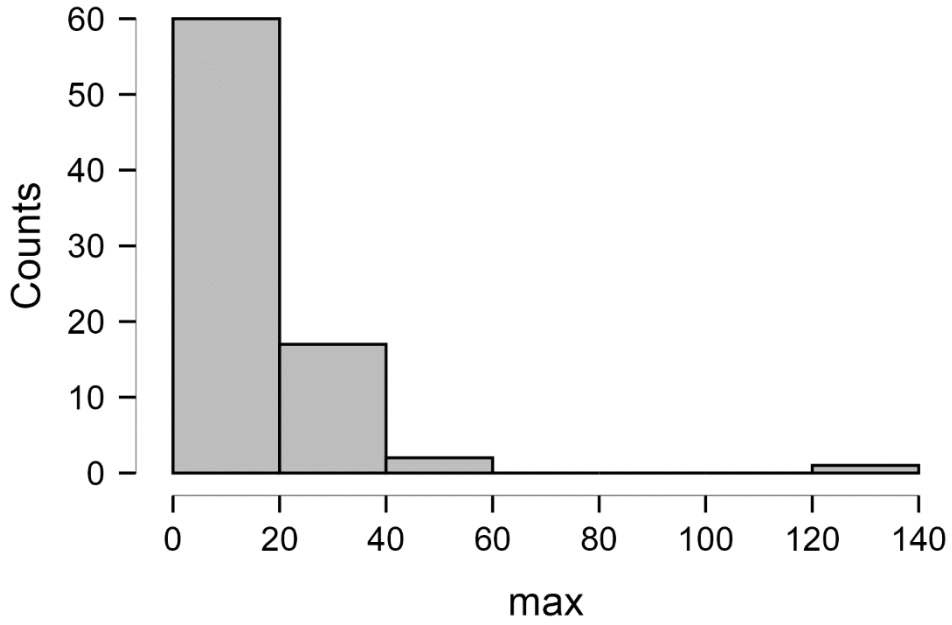
Σχήμα 24-Distribution Pie Μετρικής 5



Σχήμα 25-Minimum Distribution Plot Μετρικής 5



Σχήμα 26-Median Distribution Plot Μετρικής 5



Σχήμα 27-Maximum Distribution Plot Μετρικής 5

Τα αποτελέσματα της μετρικής 5 είναι αρκετά παρόμοια με αυτά της μετρικής 3, καθώς όπως στην μετρική 3 έτσι και στην μετρική 5 το 97.5 των έργων έχει τουλάχιστον 1 ζευγάρι διαδοχικών εκδόσεων στο οποίο πλέον δεν υπάρχουν καθόλου αλλαγές δηλαδή ούτε προστέθηκαν πακέτα ούτε αφαιρέθηκαν. Παράλληλα βλέπουμε ότι τα Distribution plots των Min και Max σχεδόν ταυτίζονται με αυτά της μετρικής 3 ενώ η κατανομή του μέσου όρου σε αυτήν την μετρική συγκλίνει δεξιότερα στον οριζόντιο άξονα. Η κατανομή αυτή του μέσου όρου ήταν αναμενόμενη και λογικό να αυξηθεί καθώς προστέθηκαν στις αλλαγές και τα πακέτα που αφαιρούνται σε αντίθεση με την μετρική 3.

Μετρική 6 - Πακέτα που προστέθηκαν και αφαιρέθηκαν σε σχέση με προηγούμενες εκδόσεις

Package Name	Counts	Total	Proportion
@11ty/eleventy-plugin-inclusive-language	1	2396	4.174e-4
@actions/core	1	2396	4.174e-4
@actions/github-4.0.0	1	2396	4.174e-4
@angular/localize	1	2396	4.174e-4
@angular/service-worker	1	2396	4.174e-4
@ava/babel	1	2396	4.174e-4
@ava/babel-preset-stage-4	1	2396	4.174e-4
@ava/babel-preset-transform-test-files	1	2396	4.174e-4
@babel/cli	5	2396	0.002
@babel/code-frame	1	2396	4.174e-4
@babel/core	15	2396	0.006
@babel/eslint-parser	4	2396	0.002
@babel/eslint-plugin	1	2396	4.174e-4
@babel/generator	1	2396	4.174e-4
@babel/node	1	2396	4.174e-4
@babel/plugin-external-helpers	2	2396	8.347e-4
@babel/plugin-proposal-class-properties	1	2396	4.174e-4
@babel/plugin-proposal-do-expressions	1	2396	4.174e-4
@babel/plugin-proposal-logical-assignment-operators	3	2396	0.001
@babel/plugin-proposal-nullish-coalescing-operator	2	2396	8.347e-4
@babel/plugin-proposal-object-rest-spread	2	2396	8.347e-4
@babel/plugin-syntax-async-generators	1	2396	4.174e-4
@babel/plugin-syntax-dynamic-import	2	2396	8.347e-4
@babel/plugin-syntax-object-rest-spread	1	2396	4.174e-4
@babel/plugin-syntax-optional-catch-binding	1	2396	4.174e-4
@babel/plugin-transform-modules-amd	2	2396	8.347e-4

Εικόνα 10-Αποτελέσματα Μετρικής 6

Package Name: Όνομα πακέτου

Counts: Πόσες φορές εντοπίστηκε αυτό το πακέτο από το συνολικό όγκο δεδομένων

Total: Συνολικός αριθμός των πακέτων που χρησιμοποιήθηκαν

Proportion: Το κομμάτι από τον συνολικό αριθμό που αντιστοιχεί σε κάθε πακέτο (e -4 σημαίνει ο αριθμός και 4 μηδενικά)

Εξαιτίας του τεράστιου όγκου δεδομένων όπως φαίνεται και στο Total της εικόνας(2396 πακέτα στο σύνολο) είναι πρακτικά αδύνατο να παρουσιαστεί όλος ο πίνακας για αυτό επιλέξαμε να παρουσιάσουμε ένα μικρό αντιπροσωπευτικό κομμάτι των αποτελεσμάτων

Από τα παραπάνω αποτελέσματα που αντλήσαμε από το JASP παρακάτω βρίσκονται τα 20 πιο δημοφιλή πακέτα που προστέθηκαν και αφαιρέθηκαν σε όλο το εύρος των 80 έργων κατά φθίνουσα σειρά.

1) Uglify-js – 25 φορές.
2) Rollup – 22 φορές.
3) Eslint– 19 φορές.
4) Babel/core – 15 φορές.
5) Browserify – 15 φορές.
6) Babel/present-env – 15 φορές.
7) Prettier – 14 φορές.
8) Typescript – 13φορές.
9) Rollup-plugin-buble – 12 φορές.
10) Rollup-plugin-commonjs – 12 φορές.
11) Rollup-plugin-node-resolve – 12 φορές.
12) Babel-core – 11 φορές.
13) Acorn – 11 φορές.
14) Grunt – 10 φορές.
15) Karma – 10 φορές.
16) Karma-chrome-launcher – 10 φορές.
17) Babel-eslint – 9 φορές.
18) Nyc – 9 φορές.
19) Requirejs – 9 φορές.
20) Webpack-cli – 9 φορές.

Μετρική 7 - Αριθμός εκδόσεων μεταξύ του πακέτου που χρησιμοποιείται στο project και την τελευταία έκδοση του πακέτου

Πίνακας 16-Βασικά Μεγέθη Μετρικής 7

	Total Packages
Valid	10829
Missing	0
Mean	34.919
Std. Deviation	105.663
Skewness	9.191
Std.Error of Skewness	0.024
Minimum	0.000
Maximum	2185.000

Valid: Ο αριθμός των έγκυρων δεδομένων που συμμετείχαν στην ανάλυση

Missing: Ο αριθμός των δεδομένων που απουσιάζουν από την ανάλυση

Mean: Μέσος Όρος των δεδομένων

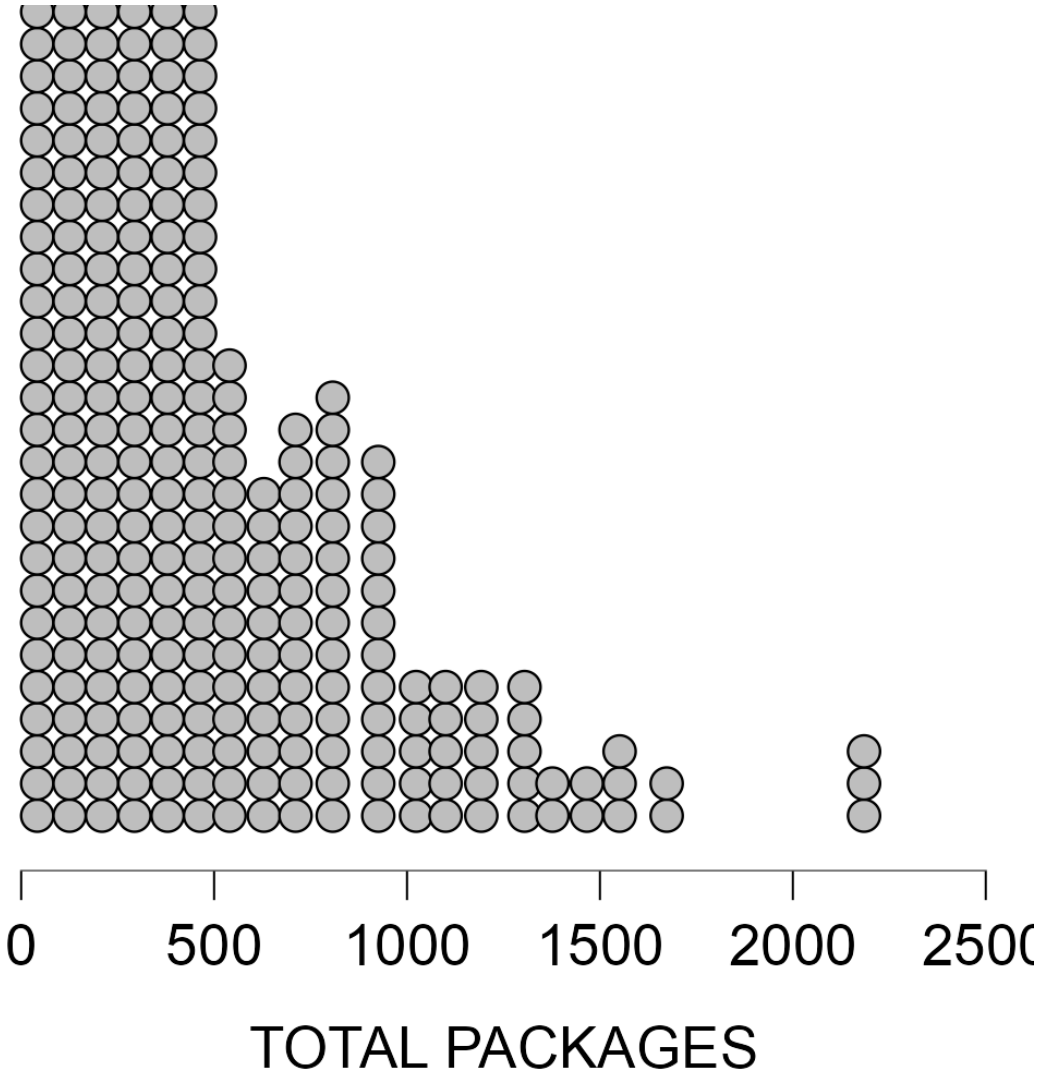
Std.Deviation: Το ποσό της διασποράς του συνόλου των δεδομένων

Skewness: Αριθμός που δείχνει πόσο ασύμμετρη είναι η κατανομή μας σε σχέση με μια κανονική κατανομή

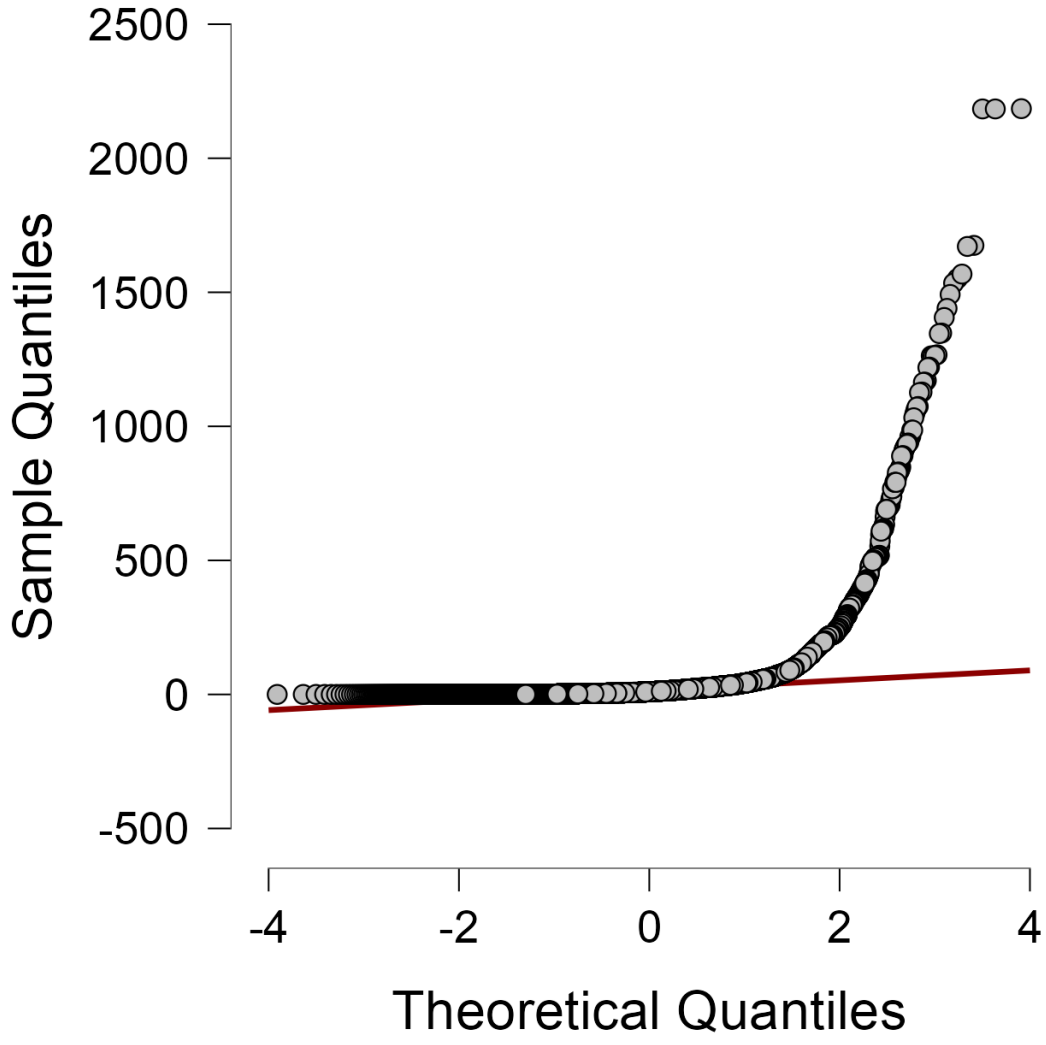
Std.Error of Skewness: Εάν αυτός ο αριθμός είναι μεγαλύτερος του +2 και μικρότερος του -2 αυτόματα καταλαβαίνουμε ότι η κατανομή μας δεν είναι κανονική

Minimum: Ελάχιστη τιμή των δεδομένων

Maximum: Μέγιστη τιμή των δεδομένων



Σχήμα 28-Total Packages Distribution Plot



σε

Σχήμα 29-Total Packages Q-Q Plot

Στα αποτελέσματα της μετρικής 7 βλέπουμε ξεκάθαρα την απουσία κανονικής κατανομής και από τα Q-Q plots και από τα Dot plots. Επίσης καταλαβαίνουμε ότι το μεγαλύτερο ποσοστό της κατανομής μας βρίσκεται μεταξύ 0-500 εκδόσεις πίσω, αποτέλεσμα το οποίο θα αναλυθεί περαιτέρω στο επόμενο κεφάλαιο.

Σε αυτό το σημείο ολοκληρώσαμε το κεφάλαιο 4 παρουσιάζοντας τα αποτελέσματα κάθε μετρικής είτε με χρήση πινάκων είτε με χρήση σχημάτων, με μοναδικό σκοπό ο αναγνώστης να είναι ικανός έχοντας εξετάσει τα αποτελέσματα να αξιολογήσει και να κρίνει τα συμπεράσματα τα οποία θα προκύψουν στο επόμενο κεφάλαιο.

Κεφάλαιο 5: Συμπεράσματα

Στο συγκεκριμένο κεφάλαιο θα αντλήσουμε τις πληροφορίες που λάβαμε από την παραπάνω έρευνα, στοχεύοντας στο να μεταφράσουμε τα στατιστικά δεδομένα με τέτοιο τρόπο που να μας δίνεται η δυνατότητα να κατανοήσουμε καλύτερα τον σκοπό αυτής της έρευνας και τέλος να καταλήξουμε σε ασφαλή συμπεράσματα για την επαναχρησιμοποίηση και εξέλιξη λογισμικού.

5.1 Οι Νόμοι του Lehman στην Σημερινή Εποχή

Αναμφισβήτητα η έρευνα του Manny Lehman την δεκαετία του 70 αφενός για τα δεδομένα εκείνης της εποχής το έργο του είχε τεράστια επιρροή και αποτέλεσε την βάση για την εξέλιξη του λογισμικού αφετέρου δημιουργείται το εξής εύλογο ερώτημα. Μπορούν οι νόμοι του Lehman να έχουν εφαρμογή στην σημερινή εποχή;

Σε αυτήν την ενότητα θα απαντήσουμε στην παραπάνω ερώτηση εξάγοντας τα δεδομένα από το προηγούμενο κεφάλαιο δημιουργώντας μια λίστα με τα συμπεράσματα που μπορούν να προκύψουν από την στατιστική ανάλυση για κάθε νόμο.

Συνεχής Αλλαγή(Continuing Change)

Ένα E-type σύστημα πρέπει διαρκώς να προσαρμόζεται αλλιώς φθίνει η χρησιμότητα του μέχρι το σημείο που είναι καλύτερο να κατασκευαστεί από την αρχή.

Η έλλειψη κανονικής κατανομής στα αποτελέσματα της μετρικής 1 φανερώνει ότι τα έργα αποτελούνται από πακέτα με παλαιότερες εκδόσεις αλλά και καινούριες εκδόσεις. Αυτό μας βοηθάει στο να καταλήξουμε ότι οι συγγραφείς έχουν την τάση να χρησιμοποιούν τις εκδόσεις των πακέτων με τις οποίες είναι εξοικειωμένοι αδιαφορώντας για τα προνόμια που προσφέρουν οι αναβαθμίσεις. Με βάση τα παραπάνω μπορεί να πει κάποιος ότι η πάροδος του χρόνου δεν φθείρει την αξιοπιστία ενός πακέτου και η συνεχής αλλαγή δεν αποτελεί πανάκεια για τους συγγραφείς της σημερινής εποχής.

Αυξανόμενη Πολυπλοκότητα(Increasing Complexity)

Κατά την διάρκεια εξέλιξης ενός συστήματος η πολυπλοκότητα του αυξάνεται εάν δεν προσπαθήσει κάποιος να το διατηρήσει ή να το μειώσει

Τα αποτελέσματα της μετρικής 2 δείχνουν ξεκάθαρα ότι η πολυπλοκότητα ενός συστήματος(έργου στην προκειμένη περίπτωση) έχει πολύ μεγάλη μεταβλητότητα στην πάροδο του χρόνου. Αντίθετα όμως δεν έχουμε καμία σημαντική ένδειξη ότι η πολυπλοκότητα αυξάνεται σε κάθε καινούρια έκδοση καθώς μπορεί είτε να αυξάνεται με την πάροδο του χρόνου είτε να μειώνεται. Ένα πιο ασφαλές συμπέρασμα είναι ότι απλούστατα η πολυπλοκότητα έχει τάση να αλλάζει γενικότερα στην πάροδο του χρόνου.

Αυτορρύθμιση(Self-Regulation)

Ένα E-type σύστημα είναι μια αυτορρυθμιζόμενη διαδικασία στην οποία τα προϊόντα και οι διαδικασίες πλησιάζουν την κανονική κατανομή

Σύμφωνα με τον παραπάνω νόμο οποιαδήποτε μέτρηση πάνω σε ένα σύστημα θα έπρεπε να ακολουθεί κανονική κατανομή δηλαδή οι μεταβλητές να έχουν την τάση να συγκεντρώνονται κοντά στον μέσο όρο. Σύμφωνα με την μετρική που επιλέξαμε και παρατηρώντας κυρίως τον μέσο όρο των έργων δεν εμφανίζεται κανονική κατανομή με αποτέλεσμα να δημιουργούνται αμφιβολίες για τον παραπάνω νόμο και τις εφαρμογές του στην σημερινή εποχή.

Διατήρηση οργανωτικής σταθερότητας(Conservation of organizational Stability)

Ο ρυθμός ανάπτυξης τείνει να είναι σταθερός, εάν δεν εφαρμοστούν κατάλληλοι μηχανισμοί για ένα E-type σύστημα

Με άλλα λόγια οι εργατικές ώρες που απαιτούνται για την αναβάθμιση ενός συστήματος είναι πάντα οι ίδιες. Το γεγονός ότι τα περισσότερα έργα έχουν τουλάχιστον μια αναβάθμιση στην οποία δεν έγινε καμία αλλαγή στα πακέτα δεν μας επιτρέπει να βγάλουμε σαφή συμπεράσματα. Στα έργα όμως που είχανε αλλαγές σε όλα τα πακέτα τους (πχ. Browserify) παρατηρούμε ότι ο νόμος απορρίπτεται καθώς το μέγιστο και το ελάχιστο έχουν διαφορετική τιμή, αντίθετα στο έργο noVNC το ελάχιστο και το μέγιστο έχουν ίδια τιμή. Μια επέκταση της ανάλυσης αυτής σε ένα μεγαλύτερο εύρος έργων JS αλλά και σε άλλες γλώσσες θα αποτελούσε σημαντική ενίσχυση για να βεβαιωθούμε για την εγκυρότητα του νόμου. Οπότε η υπόθεση απορρίπτεται.

Διατήρηση οικειότητας(Conservation of Familiarity)

Καθώς ένα σύστημα εξελίσσεται όλοι όσοι εμπλέκονται σε αυτό προγραμματιστές, πωλητές χρήστες κ.α. πρέπει να παραμένουν οικείοι με το περιεχόμενο διότι το σύστημα δεν μπορεί να έχει τεράστιες αλλαγές μεταξύ των εκδόσεων και οι αλλαγές να εξελίσσονται με σταθερό ρυθμό ώστε το σύστημα να είναι παραγωγικό

Τα σημαντικότερα συμπεράσματα που μπορούμε να εξάγουμε από την μετρική που επιλέξαμε είναι ότι το 97.5% των έργων έχει τουλάχιστον μία αναβάθμιση με μηδενικές αλλαγές όσον αφορά τα επαναχρησιμοποιούμενα πακέτα. Επίσης κατά μέσο όρο προσθαφαιρούν σχεδόν τρία πακέτα. Τέλος δεν έχουμε κάποια ένδειξη ότι η ανάπτυξη γίνεται με σταθερό ρυθμό αλλά είναι δεδομένο ότι οι προγραμματιστές δεν προτιμάνε μεγάλες αλλαγές στις αναβαθμίσεις τους για να μην υπάρχει το ρίσκο να χάσουν τον έλεγχο του συστήματος.

Συνεχής Ανάπτυξη(Continuing Growth)

Οι λειτουργίες του συστήματος πρέπει διαρκώς να αυξάνονται σε όλη την διάρκεια ζωής του με σκοπό την ικανοποίηση των χρηστών

Με την βοήθεια των προηγούμενων μετρικών και παράλληλα με τα αποτελέσματα της μετρικής 6 καταλήγουμε στο συμπέρασμα ότι παρόλο που οι εκδότες έχουν την τάση να μην ρισκάρουν πολλές αλλαγές στις αναβαθμίσεις όμως το ογκώδες εύρος διαφορετικών πακέτων μας υποδεικνύει ότι είναι πρόθυμοι να πειραματιστούν με καινούριες λειτουργίες ώστε να ανταπεξέλθουν στις απαιτήσεις των καταναλωτών.

Φθίνουσα ποιότητα(Declining Quality)

Η ποιότητα ενός συστήματος τείνει να μειώνεται εκτός εάν προσαρμόζεται στις αλλαγές του περιβάλλοντος

Από την πρώτη μετρική και τον νόμο της συνεχούς αλλαγής έχουμε συμπεράνει ότι οι εκδότες προτιμάνε εκδόσεις πακέτων με τις οποίες είναι εξοικειωμένοι, έτσι και στην μετρική 7 βλέπουμε ότι κατά μέσο όρο οι προγραμματιστές χρησιμοποιούν σχεδόν τριανταπέντε εκδόσεις πίσω από την τελευταία έκδοση, αριθμός ικανός να κάνει πιο ξεκάθαρο το γεγονός ότι τελικά οι εκδότες/προγραμματιστές καταφεύγουν σε παλαιότερες εκδόσεις πακέτων για επαναχρησιμοποίηση.

Πίνακας 17-Πίνακας Απόρριψης των Νόμων

Τίτλος Νόμων	**Αποδοχή/Απόρριψη**
Συνεχής Αλλαγή	X
Αυξανόμενη Πολυπλοκότητα	X
Αυτορρύθμιση	X
Διατήρηση οργανωτικής σταθερότητας	X
Διατήρηση Οικειότητας	√
Συνεχής Ανάπτυξη	√
Φθίνουσα Ποιότητα	X

-Το σύμβολο '√' οι νόμοι που επιβεβαιώνονται.

-Το σύμβολο 'X' όσοι απορρίπτονται

5.2 Νόμοι του Lehman Μέσα από Άλλες Έρευνες

Σε αυτήν την ενότητα θα γίνει μια απόπειρα σύγκρισης των συμπερασμάτων που προέκυψαν στην παραπάνω ενότητα με εγκεκριμένες έρευνες για την εφαρμογή των νόμων του Lehman στην σημερινή εποχή.

Στον πίνακα 8 φαίνονται τα συμπεράσματα που προέκυψαν από προηγούμενες έρευνες συναρτήσει με αυτήν την έρευνα. Παρουσιάζονται 9 διαφορετικές μελέτες [9],[10],[11],[12],[13],[14],[15],[16],[17], σε διάφορες γλώσσες προγραμματισμού ταξινομημένες με τυχαία σειρά καθώς και τα αποτελέσματα που προέκυψαν για τους 8 νόμους του Lehman και την εγκυρότητα τους στην εξέλιξη λογισμικού. Όπως μπορεί να παρατηρήσει κανείς τα αποτελέσματα αυτά κατά γενική ομολογία έχουν διαφορές μεταξύ τους. Ωστόσο δεν μπορούμε να παραλείψουμε το γεγονός ότι ο πρώτος νόμος επιβεβαιώνεται από όλους τους προηγούμενους ερευνητές ενώ εμείς τον απορρίψαμε. Αντίθετα όπως φαίνεται και στον πίνακα 8 ο έκτος νόμος επιβεβαιώνεται από όλες τις έρευνες που αναφέρονται. Με άλλα λόγια όλοι οι ερευνητές συμφωνούν για την Συνεχή Ανάπτυξη του λογισμικού με σκοπό το σύστημα να παραμένει ανταγωνιστικό στην αγορά. Τέλος παρατηρούμε ότι μόνο οι [14] επιβεβαιώνουν ότι το σύστημα πρέπει να είναι ανατροφοδοτούμενο ενώ η [11] είναι η μοναδική έρευνα που επιβεβαιώνει την Φθίνουσα Ποιότητα ενός συστήματος.

Πίνακας 18-Σύγκριση Αποτελεσμάτων με Άλλες Έρευνες

Αναφορές	**Έτος**	**Γλώσσα**	**I**	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**
Xie et al	2009	C	√	√	√	~	X	√	X	X
Αμανατίδης& Χατζηγεωργίου	2015	PHP	√	X	√	√	√	√	~	X
Kaur et al	2014	C++	√	√	√	~	√	√	√	~
Neamtiu et al	2013	C	√	X	X	X	X	√	X	X
Businge et al	2010	Java	√		√		X	√	~	
Israeli&Feitelson	2010	C	√	X	√	√	~	√	X	√
Mens et al	2008	Java	√	X				√		
Robles et al	2005	C, C++,Java	√			X		√		X
Godfrey & Tu*	2000	C	√			X		√		X
Αυτή η έρευνα	2022	JS	X	X	X	X	√	√	X	

-Το σύμβολο '√' οι νόμοι που επιβεβαιώνονται.

-Το σύμβολο '~ ' οι απροσδιόριστοι

-Το σύμβολο 'X ' όσοι απορρίπτονται

-Το κενό σημαίνει ότι δεν μελετήθηκε ο νόμος

5.3 Μελλοντική Επέκταση

Ολοκληρώνοντας και αναλύοντας την συγκεκριμένη έρευνα δημιουργείτε η ευκαιρία για περαιτέρω έρευνα και επέκταση επί του θέματος. Το ιδανικότερο σενάριο θα ήταν να υλοποιήσουμε μια έρευνα με μεγαλύτερο όγκο δεδομένων, περισσότερα έργα αλλά και περισσότερες συγκρίσεις μεταξύ των αναβαθμίσεων με σκοπό να μπορούμε με περισσότερη σιγουριά και αυτοπεποίθηση να εξάγουμε τα συμπεράσματα μας. Εξίσου σημαντική είναι η δημιουργία νέων μετρικών που θα βοηθήσουν να συλλέξουμε νέα δεδομένα και θα ανοίξουν τον δρόμο για μια πιο εμπεριστατωμένη ερεύνα στην εγκυρότητα των νόμων του Lehman στην σημερινή εποχή. Επιπρόσθετα θα επιθυμούσαμε να διαμορφώσουμε μια μετρική για την συλλογή στοιχείων που θα βοηθήσουν να επεξεργαστούμε και να αναλύσουμε και τον τελευταίο νόμο του Lehman, Ανατροφοδοτούμενο Σύστημα(Feedback System) για τον οποίο δεν εφαρμόστηκε μετρική στην παραπάνω έρευνα. Μία επιπλέον καλή ιδέα θα ήταν να αξιοποιήσουμε τα δεδομένα που συλλέξαμε στην παραπάνω μελέτη κατασκευάζοντας μια πιο εξειδικευμένη στατιστική ανάλυση προεκτείνοντας την περιγραφική στατιστική που εφαρμόστηκε σε επαγωγική στατιστική με σκοπό να προβλέψουμε καλύτερα την εξέλιξη των έργων JavaScriptκαι την επαναχρησιμοποίηση λογισμικού.

Κεφάλαιο 6: Επίλογος

6.1 Εγκυρότητα και Αξιοπιστία

Στην συγκεκριμένη ενότητα θα επιχειρήσουμε να εξετάσουμε την εγκυρότητα και την αξιοπιστία αυτής της μελέτης. Αρχικά η μετάφραση των νόμων από το θεωρητικό κομμάτι στο πρακτικό και την δημιουργία μετρικών είναι καθαρά υποκειμενική και ενδέχεται να διαφέρει από ερευνητή σε ερευνητή οπότε είναι δεδομένο ότι βλάπτει την επιφανειακή εγκυρότητα(Face Validity) και την εγκυρότητα κατασκευής (Construct Validity) της έρευνας. Επίσης είναι απαραίτητο να προσθέσουμε ότι η εξωτερική εγκυρότητα της έρευνας επηρεάζεται καθώς εφαρμόσαμε φίλτρα ταξινόμησης στο αποθετήριο GitHub για την επιλογή των έργων που επιλέχθηκαν για μελέτη όπως αναφέρθηκε και στο κεφάλαιο 2 και 3 με την επιλογή να γίνεται τυχαία και σίγουρα δεν αντικατοπτρίζουν όλο το πεδίο του επαναχρησιμοποιημένου λογισμικού σε γλώσσα JavaScript. Έτσι η μελέτη διαφορετικών έργων θα αποτελέσει χρήσιμο εργαλείο για την επιστημονική κοινότητα. Όσον αφορά την εσωτερική εγκυρότητα δεν έγινε χρήση αιτιατής υπόθεσης οπότε δεν υπάρχει κίνδυνος για αλλοίωση αποτελεσμάτων. Υποκειμενικότητα επίσης εφαρμόστηκε κατά την σύγκριση των αναβαθμίσεων μεταξύ τους διότι η υπολογιστική δύναμη που διαθέταμε δεν ήταν αρκετή για να καλύψει όλο το εύρος των αναβαθμίσεων αλλά η δημιουργία μιας μεθόδου σύγκρισης και η συνέπεια σε αυτήν θεωρούμε ότι δεν βλάπτουν την αξιοπιστία της μελέτης. Τέλος καθώς αυτή η μελέτη βασίζεται κυρίως στην ερμηνεία στατιστικών αποτελεσμάτων γεννιούνται απειλές για την συμπερασματική εγκυρότητα(Conclusion Validity) της έρευνας.

6.2 Ωφελιμότητα του Έργου

Παρόλο που τα αποτελέσματα αυτής της έρευνας βασίζονται περισσότερο στο θεωρητικό κομμάτι της εγκυρότητας των νόμων του Lehman στην σημερινή εποχή δεν αποκλείεται το γεγονός να φανούν χρήσιμα και στο πρακτικό κομμάτι. Ως ακολούθως οι ερευνητές και οι προγραμματιστές μπορούν να αντλήσουν ωφέλιμες πληροφορίες από την μελέτη αυτήν.

-Σύμφωνα με τα αποτελέσματα που συμπεράναμε παραπάνω δεν παρατηρείται αύξηση της πολυπλοκότητας του συστήματος καθώς αναπτύσσεται. Αυτό μπορεί να μεταφραστεί πρακτικά ότι μια ομάδα προγραμματιστών η οποία βλέπει αύξηση της πολυπλοκότητας του έργου κατά την ανάπτυξη του μέσα στον χρόνο ίσως είναι ένας παράγοντας στον οποίο πρέπει να δώσουν την κατάλληλη προσοχή ώστε να καταφέρουν να αυξήσουν την αποτελεσματικότητα τους.

-Η επιβεβαίωση του νόμου ότι οι προγραμματιστές και όσοι εμπλέκονται στην δημιουργία ενός έργου πρέπει να μένουν οικείοι με αυτό μπορεί να δώσει πάτημα στους προγραμματιστές να καταλάβουν ότι οι μεγάλες αλλαγές μπορεί να επιφέρουν προβλήματα στην κατανόηση του. Αντίθετα μικρές και στοχευμένες αλλαγές είναι αυτό που πρέπει να επιδιώκουν.

Η υπόθεση ότι τα έργα συνεχώς αναπτύσσονται επιβεβαιώνεται και από αυτήν την έρευνα αλλά και από τις προηγούμενες. Ως εκ τούτου μια άμεση πληροφορία που προκύπτει είναι ότι οι project managers θα πρέπει να είναι προετοιμασμένοι να ανταπεξέλθουν στις αυξανόμενες ανάγκες για πόρους που θα προκύψουν ώστε να μπορέσουν να διατηρήσουν το έργο τους και να παραμείνουν ανταγωνιστικοί.

Όπως τονίσαμε και παραπάνω, η μελέτη αυτή έχει χρησιμότητα και στους ερευνητές οι οποίοι μέσα από αυτήν την έρευνα μπορούν να αντλήσουν τις δικές τους πληροφορίες από τα αποτελέσματα που προέκυψαν.

-Οι ερευνητές μπορούν να αντλήσουν πληροφορίες από τις μετρικές που εφαρμόσαμε αλλά και την διαδικασία στην οποία βασίστηκε αυτή η έρευνα

-Τους ενθαρρύνουμε να πειραματιστούν με το JASP ως ένα εναλλακτικό εργαλείο για στατιστική ανάλυση.

-Ακόμη μπορεί να φανεί χρήσιμος ο κώδικας και οι εντολές που χρησιμοποιήθηκαν για την συλλογή των δεδομένων αυτής της μελέτης

-Τέλος παροτρύνουμε τους ερευνητές να κοιτάξουν το εγχείρημα αυτό σαν μια εμπειρική έρευνα πάνω στην εγκυρότητα των νόμων του Lehman στην επαναχρησιμοποίηση λογισμικού.

Παραρτήματα

Στο κεφάλαιο παραρτήματα μας δίνεται η ευκαιρία να παρουσιάσουμε τα εργαλεία και τον κώδικα που χρησιμοποιήθηκε για να αυτοματοποιηθεί ολόκληρη η διαδικασία που χρησιμοποιήθηκε για την έρευνα αυτήν και την επίτευξη των στόχων μας

Κώδικας BASH

Πρώτο Αρχείο Κώδικα

Η παρουσίαση του κώδικα θα γίνει με ανάλυση με την σειρά της διαδικασίας εκτέλεσης του κώδικα, επεξήγηση των σημαντικότερων εντολών και επίδειξη εικόνων.

1. ***Λήψη του αρχείου releases από το API του GitHub***

Στο αρχείο αυτό βρίσκονται οι 30 τελευταίες εκδόσεις του έργου τις οποίες θα χρησιμοποιήσουμε.

```
#!/bin/bash
start=$SECONDS
if [ $# != 2 ];then
        echo "I need 2 variables"
        exit
fi
wget -r -np -l1 https://api.github.com/repos/$1/$2/releases
cd /mnt/c/Users/30695/Desktop/api.github.com/repos/$1/$2

grep -e  "tag_name" -e "published_at" releases  | cut -d '"' -f4 >$2-release.txt

sed -i 's/[,]//g' $2-release.txt
sed -i 's/["]//g' $2-release.txt
tr '\n' '+' < $2-release.txt  > $2_release.txt
sed -i 's/Z+/Z\n/g' $2_release.txt
rm releases
```

Εικόνα 11-Εκτέλεση 1 του Κώδικα

- Χρήση εντολής if για και exit για να δώσει ο χρήστης 2 παραμέτρους ώστε να μπορεί να προχωρήσει το πρόγραμμα αλλιώς τερματίζει.
- Χρήση εντολής wget για να κατεβάσουμε το αρχείο releases από το API του GitHub
- Χρήση εντολής grep, sed και tr για φιλτράρισμα με σκοπό στο αρχείο να απομείνουν μόνο ο αριθμός της έκδοσης
- Χρήση εντολής rm για διαγραφή περιττών αρχείων.

2. *Λήψη των συμπιεσμένων αρχείων των εκδόσεων και αποσυμπίεση στον κατάλληλο φάκελο*

Στα συμπιεσμένα αρχεία είναι οι εκδόσεις του έργου που περιέχουν μέσα τα αρχεία package.json

```
input="/mnt/c/Users/30695/Desktop/api.github.com/repos/$1/$2/$2_release.txt"
while IFS= read -r line
do
        zip=$( echo "$line" | cut -d '+' -f1)
        echo $zip
                wget  -r -np -l1 https://github.com/$1/$2/archive/tags/$zip".zip"
        done < "$input"



        cd /mnt/c/Users/30695/Desktop/TAGS
        mkdir -p $1

        cd /mnt/c/Users/30695/Desktop/api.github.com/repos/$1/$2/github.com/$1/$2/archive

        mv tags  /mnt/c/Users/30695/Desktop/TAGS/$1
        cd /mnt/c/Users/30695/Desktop/json
        mkdir -p $2
        cd /mnt/c/Users/30695/Desktop/TAGS/$1/tags
        unzip -q "*.zip" -d /mnt/c/Users/30695/Desktop/json/$2
        cd /mnt/c/Users/30695/Desktop/json/$2
```

Εικόνα 12-Εκτέλεση 2 του Κώδικα

- Χρήση input για να εντοπίσει το αρχείο release που χρειαζόμαστε
- Χρήση βρόγχου με την εντολή while και do για να διαβάσει όλες τις σειρές του αρχείου releases
- Χρήση echo και cut για την αποθήκευση των εκδόσεων στην μεταβλητή zip
- Χρήση wget για λήψη των συμπιεσμένων αρχείων
- Χρήση unzip για αποσυμπίεση στον φάκελο json

3. *Διαγραφή των αρχείων που δεν είναι package.json και μεταφορά τους σε κεντρικό κατάλογο μετά από μετονομασία.*

```
        cd /mnt/c/Users/30695/Desktop/json/$2

        find . -not -name "package.json" -type f -delete
        find . -type d -empty -delete

        counter=0



        for f in *; do [[ -d "$f" ]] && {
                dir=$f
        echo $dir
        cd $dir
        s=1
        find .  -name "package*.json"  -type f | while read f
do
        newname="package$s.json"
        mv -n "$f" "$newname"

        s=$((s+1))
done

cd ..
}; done
```

Εικόνα 13-Εκτέλεση 3 του Κώδικα

- Χρήση εντολής find για εντοπισμό των αρχείων που δεν είναι package.json και διαγραφή
- Χρήση εντολής for για ανάγνωση όλων των αρχείων
- Χρήση μεταβλητών για την μετονομασία και αρίθμηση όλων των αρχείων package.json
- Χρήση εντολής mv για μεταφορά στον κεντρικό κατάλογο

4. *Φιλτράρισμα των αρχείων package.json*

Διαγραφή των ειδικών χαρακτήρων και τον κενών σειρών από τα αρχεία package.json για ευκολία στην ανάγνωση.

```
find .  -name "package*.json"  -type f  | while read f
do

        touch $f.txt
        sed -i  's|[",{, ~, \^ ]||g' $f >>$f.txt
        sed -i  '/^[[:space:]]*$/d' $f >> $f.txt
        sed -i  's/[[:blank:]]//g' $f >> $f.txt
        sed -i  's|[    ]||g'  $f >> $f.txt
        sed -i  's|[            ]||g'  $f >> $f.txt
        sed -i  's|[ ]||g'  $f >> $f.txt
        sed -i  's/\"/\\\"/g' $f >> $f.txt
        sed -i  's/[][]//g' $f >> $f.txt
        sed -i  's/https:/https/' $f >> $f.txt
        sed -i  's/http:/http/' $f >> $f.txt
        sed -i  's/github:/github-/' $f >> $f.txt
        sed -i  's/:workspace/-workspace/' $f >> $f.txt
        sed -i  's/ERROR:/ERROR=/' $f >> $f.txt
        sed -i  's/Error:/ERROR=/' $f >> $f.txt
        sed -i  's/link:/link=/' $f >> $f.txt
        sed -i  's/git:/git=/' $f >> $f.tx
        sed -i  's/npm:/npm=/' $f >> $f.txt
        sed -i  's/ssh:/ssh=/' $f >> $f.txt
        sed -i  's/gist:/gist=/' $f >> $f.txt
        awk '{$1=$1};1'  $f >> $f.txt
        tr "\011" "*"< $f > $f.txt
        tr -d "      " < $f > $f.txt
        tr -d "[]" < $f > $f.txt
done
```

Εικόνα 14-Εκτέλεση 4 του Κώδικα

- Χρήση εντολής find για εντοπισμό των αρχείων package.json
- Χρήση εντολής touch για δημιουργία αρχείου .txt
- Χρήση εντολών sed, awk, και tr για διαγραφή και τροποποίηση περιττών χαρακτήρων

5. *Διαχωρισμός devDependecies και dependencies από τα αρχεία package.json*

```
for a in *; do [[ -d "$a" ]] && {
        m=$(echo "$a" | cut -d '-' -f 3)
echo $m
cd $a
test200=$2"-"$m"-.txt"
touch $test200
echo $test200
find .  -name "package*.json"  -type f | while read packe
do
        touch $packe.txt
        awk '/devDependencies/{flag=1; next;} /}/{flag=0} flag' $packe |tr ':' ','>> $packe.txt
        grep -o 'devDependencies.* }' $packe |tr ':' ','>> $packe.txt

        while IFS=$\r read -r line2 ;do
                echo "$line2,$m">>$test200
        done < $packe.txt
        rm $packe.txt

        touch $packe.txt
        awk '/dependencies/{flag=1; next;} /}/{flag=0} flag' $packe |tr ':' ','>> $packe.txt
        grep -o 'dependencies.* }'  $packe |tr ':' ','>> $packe.txt
        while IFS=$\r read -r line3 ;do
                echo "$line3,$m">>$test200
        done < $packe.txt
        rm $packe.txt
done
cd ..
}; done
```

Εικόνα 15-Εκτέλεση 5 του Κώδικα

- Χρήση εντολής find για εντοπισμό των αρχείων package.json
- Χρήση εντολής touch για δημιουργία αρχείου
- Χρήση grep, awk, tr για φιλτράρισμα των αρχείων
- Χρήση βρόγχου do while για ανάγνωση όλων των γραμμών και αποθήκευση σε τελικό αρχείο

6. *Αποκοπή ημερομηνίας από το αρχεία releases για την δημιουργία των packunique αρχείων*

```
cd /mnt/c/Users/30695/Desktop/api.github.com/repos/$1/$2
mv $2_release.txt /mnt/c/Users/30695/Desktop/json/$2

cd /mnt/c/Users/30695/Desktop/json/$2

find .  -name "$2-*.txt"  -type f | while read packe
do
echo $packe
dateme=$( echo "$packe" | sed 's/.*'$2'-//' |  sed 's/.txt//' | tr -d '-')
echo $dateme
p=$( awk '/'"$dateme+"'/,/'"T"'/' "$2_release.txt" | cut -d '+' -f 2)
echo $p
while IFS=$\r read -r lineme ;do
echo "$lineme,$p" >> temp.txt
done < $packe
rm $packe
while IFS=$\r read -r lineme ;do
echo "$lineme" >> $packe
done < temp.txt
rm temp.txt
done
```

Εικόνα 16-Εκτέλεση 6 του Κώδικα

- Χρήση εντολής mv για μεταφορά του αρχείου release στον κατάλληλο φάκελο
- Χρήση εντολής find για εντοπισμό των αρχείων με το αντίστοιχο όνομα του έργου
- Χρήση sed, tr, awk για φιλτράρισμα του αρχείου releases και αποκοπή της ημερομηνίας
- Χρήση βρόγχου do while για εκτύπωση του ονόματος πακέτου, έκδοσης πακέτου και ημερομηνία στα αντίστοιχα αρχεία
- Χρήση rm για διαγραφή περιττών αρχείων

7. *Δημιουργία αρχείων packunique, sort και αρχεία μετρικής 2*

```
cd /mnt/c/Users/30695/Desktop/json/$2

counter="$2_total_countofpackages_perversion.txt"
touch $counter
find .  -name "$2-*.txt"  -type f | while read packe
do

        mine=$( echo "$packe" | cut -d '-' -f 1-4 |  cut -d '.' -f 1-5 |  cut -d '/' -f 2)
        echo $mine
        test300="packunique-"$mine".txt"
        touch $test300
        sort $packe >> "sort-"$mine".txt"
        sort -u -t, -k1,1 "sort-"$mine".txt" >> $test300
        wc -l  $test300 >> $counter

        rm "sort-"$mine".txt"
        sort  -u -t, -k1,1 $packe | cut -d ',' -f 1  >> "sort-"$mine".txt"


 mv "sort-"$mine".txt" /mnt/c/Users/30695/Desktop/compare/$2
        mv $test300 /mnt/c/Users/30695/Desktop/compare/$2
done
```

Εικόνα 17-Εκτέλεση 7 του Κώδικα

- Χρήση εντολής counter για δημιουργία αρχείου
- Χρήση εντολής cut για δημιουργία των αρχείων packunique
- Χρήση εντολής sort για ταξινόμηση των αρχείων και εκτύπωση στα αρχεία sort
- Χρήση εντολής wc για την καταμέτρηση των γραμμών για την μετρική 2
- Χρήση εντολής rm και mv για διαγραφή και μεταφορά αρχείων

8. *Δημιουργία αρχείων μετρικής 7*

```
cd /mnt/c/Users/30695/Desktop/compare/$2
echo Package name,Current Versions,Used Version,Project name,Project Version,Number of versions behind: >> $2_versions_behind.txt
find .  -name "packunique-*" -type f | while read packe
do
while IFS=$\r read -r line1;do
name=$(echo "$line1" | cut -d ',' -f1)
pavers=$(echo "$line1" | cut -d ',' -f2)
prvers=$(echo "$line1" | cut -d ',' -f3)
curl https://www.npmjs.com/package/$name  -o t.txt
grep -o 'versionsDownloads.*.starAction' t.txt  | tr -d '"'| tr ',' '\n' | tr -d '{' | tr -d '}'| tr -d 'versionsDownloads'| tr -d 'starAction' > tr.txt
rm t.txt
while IFS=$\r read -r line3 ;do
               vers=$( echo "$line3" | cut -d ':' -f 1)
                       echo $vers >> t.txt
done < tr.txt
sort -r t.txt >> t1.txt
rm t.txt
rm tr.txt
sed '/'"$pavers"'/q' t1.txt >> tr.txt
rm t1.txt
find  .  -name "tr.txt" -type f
cvers=$( head -1 "tr.txt")
q=$(wc -l < tr.txt)
Diff=$(( q - 1 ))
echo $cvers
echo $Diff
echo $prvers
echo $pavers
echo $name,$cvers,$pavers,$2,$prvers,$Diff >> $2_versions_behind.txt
done <$packe
done
rm tr.txt
cd /mnt/c/Users/30695/Desktop/VERSIONS-BEHIND
mkdir -p $2
cd /mnt/c/Users/30695/Desktop/compare/$2
mv $2_versions_behind.txt /mnt/c/Users/30695/Desktop/VERSIONS-BEHIND/$2
```

Εικόνα 18-Εκτέλεση 8 του Κώδικα

- Χρήση εντολής echo για εκτύπωση στα αρχεία της μετρικής 7
- Χρήση εντολής find για εντοπισμό των αρχείων packunique
- Χρήση βρόγχου για ανάγνωση όλων των γραμμών του αρχείου
- Χρήση εντολής cut για αποκοπή των ζητούμενων στοιχειών και αποθήκευση σε μεταβλητές
- Χρήση εντολής curl για λήψη των πακέτων από το npm.js
- Χρήση εντολών grep, tr για φιλτράρισμα των αρχείων που κατεβάσαμε
- Χρήση εντολής head για ανάγνωση μόνο της πρώτης γραμμής και αποθήκευση σε μεταβλητή
- Χρήση wc για καταμέτρηση των γραμμών

Δεύτερο Αρχείο Κώδικα

I. Δημιουργία αρχείων μετρικών 3,5,6

```
#!/bin/bash

echo "Just remember the second argument should be the newest"
if [ $# != 6 ];then
echo "Feed me just right. I need dir file and 2 versions "
exit
fi

cd /c/Users/admin/Desktop/compare/$2

first=$( echo "$3" | cut -d '-' -f2-5  |  cut -d '.' -f 1-4 )
sec=$( echo "$4" | cut -d '-' -f 2-5 |  cut -d '.' -f 1-4 )

third=$( echo "$5" | cut -d '-' -f 2-4 |  cut -d '.' -f 1-4 )
forth=$( echo "$6" | cut -d '-' -f 2-4 |  cut -d '.' -f 1-4 )

diff -uB $3 $4 >"different"$first"-"$sec".txt"
echo `grep "^-" "different"$first"-"$sec".txt" | grep -v "^---" | wc -l` "lines removed">> "numdiff_u_"$first"-"$sec".txt"
echo `grep "^+" "different"$first"-"$sec".txt" | grep -v "^+++" | wc -l` "lines added" >> "numdiff_u_"$first"-"$sec".txt"
echo `grep "^+" "different"$first"-"$sec".txt" | grep -v "^+++" | wc -l` "lines added" >> "lines_added_in"$first"-"$sec".txt"
removes=$(echo `grep  "^-" "different"$first"-"$sec".txt" | grep -v "^---" | wc -l`)
adds=$(echo `grep  "^+" "different"$first"-"$sec".txt" | grep -v "^+++" |   wc -l`)
echo $removes
echo $adds
echo $((adds + removes)) $first"-"$sec >> totaladds-$first-$sec.txt


diff $3 $4 > "diff_plain"$first"-"$sec".txt"
echo "removed" >> "diff_beetween_"$first"-"$sec".txt"
echo `grep "^<" diff_plain$first-$sec.txt | grep -v "^---" | sed 's/</\r/g'`>> "diff_beetween_"$first"-"$sec".txt"
echo "******************************" >>"diff_beetween_"$first"-"$sec".txt"
echo "added" >> "diff_beetween_"$first"-"$sec".txt"
echo `grep "^>"  diff_plain$first-$sec.txt | grep -v "^+++" | sed 's/>/\r/g'`  >> "diff_beetween_"$first"-"$sec".txt"



echo "same packages" >> "same_"$first"-"$sec".txt"

awk -F ',' 'NR==FNR{++a[$1];next} $1 in a'   $5 $6 >> "check_"$first"-"$sec".txt"
awk -F ',' 'NR==FNR{++a[$1];next} $1 in a'  $6  $5 >> "check_"$sec"-"$first".txt"



cat "check_"$sec"-"$first".txt" "check_"$first"-"$sec".txt"  >> temp.txt
sort  -t, -k1,1 temp.txt >> "same_"$first"-"$sec".txt"
sort -u -t, -k1,1 -k2,2 temp.txt >> "same2_"$first"-"$sec".txt"
rm temp.txt
find . -name "diff_plain*"  -type f -delete
find . -name "different*"  -type f -delete
find . -name "same_*"  -type f -delete
mv check_*.txt /c/Users/admin/Desktop
mv same2_*.txt /c/Users/admin/Desktop
cd /c/Users/admin/Desktop
```

Εικόνα 19-Εκτέλεση 9 του Κώδικα

- Χρήση μεταβλητών και εντολής cut για αποκοπή των ζητούμενων από τα αρχεία packunique.txt
- Χρήση της εντολής grep για την δημιουργία των αρχείων της μετρικής 3
- Ίδια διαδικασία και για την δημιουργία των αρχείων της μετρικής 5
- Χρήση εντολής grep και sed για την δημιουργία των αρχείων της μετρικής 6
- Χρήση της εντολής awk για την δημιουργία των αρχείων check
- Χρήση των εντολών cat και sort με παράμετρο -u για μοναδικότητα στα αρχεία check και δημιουργία των αρχείων same2
- Χρήση εντολών find,mv,rm για εύρεση, μετακίνηση και διαγραφή των αρχείων

II. Δημιουργία αρχείων μετρικής 1

```
echo Package,Package Version,Package Date,Project Version and Date, Days Between Package Update: >> final_"$first"-"$sec".txt
find .  -name "same2_"$first"-"$sec".txt"  -type f | while read packe1
do
        while IFS=$\r read -r line3;do
                vers=$(echo "$line3" | cut -d ',' -f 2)
                name=$(echo "$line3" | cut -d ',' -f 1 )

                rest=$(echo "$line3" | cut -d ',' -f 3-4 )

cd /c/Users/admin/Desktop/versions
mkdir -p $name
cd /c/Users/admin/Desktop/versions/$name
curl   https://www.npmjs.com/package/$name/v/$vers -O

 cd ..
done  < $packe1

done
cd /c/Users/admin/Desktop/versions

for f in *; do [[ -d "$f" ]] && {


cd $f
find . -type f -exec grep -Iq . {} \; -print | while read file
do


result=$(echo $file | grep -o "<time.*time>" $file  |   grep -o "=.*title"  | tr -d 'title' | tr -d '"' | tr -d '=' | grep -o '.*T' | tr -d 'T')
 echo $result >> $file.txt
rm $file
done
cd ..
};done
cd /c/Users/admin/Desktop
find .  -name "same2_"$first"-"$sec".txt"  -type f | while read packe1
do
                while IFS=$\r read -r line3;do

vers=$(echo "$line3" | cut -d ',' -f 2)
name=$(echo "$line3" | cut -d ',' -f 1 )
rest=$(echo "$line3" | cut -d ',' -f 3-4 )
date1=$(echo "$line3" | cut -d ',' -f4 )



cd /c/Users/admin/Desktop/versions/$name
find .  -name "$vers.txt"  -type f | while read packe
do
while IFS=$\r read -r line5;do


date=$(echo "$line5" |  cut -d ' ' -f1 )
DATEfirstnum=`date -d "$date" +"%y"`
DATElastnum=`date -d "$date1"  +"%y"`
DAYSdif=$(( 10#$DATElastnum - 10#$DATEfirstnum))
DATEthird=`date -d "$date" +"%j"`
DATEfour=`date -d "$date1" +"%j"`
DAYSdif2=$(( 10#$DATEfour - 10#$DATEthird  ))
DAYSdif3=$(echo "$DAYSdif2" | tr -d '-')

 cd /c/Users/admin/Desktop
 echo $name,$vers,$date,$rest,$DAYSdif years and $DAYSdif3 days >> final_"$first"-"$sec".txt

done <$packe
done
```

Εικόνα 20-Εκτέλεση 10 του Κώδικα

- Χρήση εντολής echo για δημιουργία τίτλου στο αρχείο final
- Χρήση βρόγχου με την εντολή do while και while IFS για ανάγνωση κάθε γραμμής των αρχείων
- Χρήση εντολής curl στην σελίδα npm.js για κατέβασμα των αρχείων για κάθε πακέτο
- Χρήση εντολής grep, tr για φιλτράρισμα των κατεβασμένων αρχείων
- Χρήση εντολής date για μετατροπή της ημερομηνίας σε χρόνια και μέρες
- Αφαίρεση των δύο ημερομηνιών για την δημιουργία των ζητουμένων της μετρικής 1
- Δημιουργία του τελικού αρχείου της μετρικής 1

III. Δημιουργία αρχείων μετρικής 4.1

```
cd /c/Users/admin/Desktop
done <$packe1
done
cd /c/Users/admin/Desktop
find .  -name "check_"$first"-"$sec".txt" -type f | while read packe
do
pdate1=$( head -1 "check_"$first"-"$sec".txt" | cut -d ',' -f4)

find .  -name "check_"$sec"-"$first".txt" -type f | while read packe
do
pdate2=$( head -1 "check_"$sec"-"$first".txt" | cut -d ',' -f4)
DATE1=`date -d "$pdate1" +"%j"`
echo $DATE1
DATE2=`date -d "$pdate2" +"%j"`
echo $DATE2
DIFFERENCE=$(( 10#$DATE1 - 10#$DATE2 ))
DIFFERENCE2=$( echo "$DIFFERENCE" | tr -d '-')

cd /c/Users/admin/Desktop/compare/$2
find .  -name "totaladds-$first-$sec.txt" -type f | while read packe
do
pvers=$( head -1 "totaladds-$first-$sec.txt" | cut -d ' ' -f1)
Div=$( echo "scale=2; $pvers / $DIFFERENCE2" | bc )
cd /c/Users/admin/Desktop
echo  $Div -"$first"-"$sec" >> division_$first-$sec.txt
done
done
done
```

Εικόνα 21-Εκτέλεση 11 του Κώδικα

- Χρήση εντολής find για εύρεση των αρχείων check
- Χρήση βρόγχων με τις εντολές do while
- Χρήση εντολής head για ανάγνωση της πρώτης σειράς των αρχείων check
- Χρήση εντολής date για μετατροπή των ημερομηνιών σε ημέρες
- Αφαίρεση ημερομηνιών και στην συνέχεια η διαίρεση
- Δημιουργία των τελικών αρχείων της μετρικής 4.1

IV. Μεταφορά των αρχείων σε κατάλληλους φακέλους

```
cd /c/Users/admin/Desktop/FINALS
mkdir -p $2
cd $2
mkdir -p "$first"-"$sec"
cd /c/Users/admin/Desktop
mv final_"$first"-"$sec".txt /c/Users/admin/Desktop/FINALS/$2/"$first"-"$sec"
rm  -rf  /c/Users/admin/Desktop/versions/*
cd /c/Users/admin/Desktop/TOTAL
mkdir -p $2
cd $2
mkdir -p "$first"-"$sec"
cd /c/Users/admin/Desktop/compare/$2
mv totaladds-$first-$sec.txt /c/Users/admin/Desktop/TOTAL/$2/$first-$sec
cd /c/Users/admin/Desktop/DIFF
mkdir -p $2
cd $2
mkdir -p "$first"-"$sec"
cd /c/Users/admin/Desktop/NUM
mkdir -p $2
cd $2
mkdir -p "$first"-"$sec"
cd /c/Users/admin/Desktop/ADDED
mkdir -p $2
cd $2
mkdir -p "$first"-"$sec"
cd /c/Users/admin/Desktop/compare/$2
mv "diff_beetween_"$first"-"$sec".txt" /c/Users/admin/Desktop/DIFF/$2/"$first"-"$sec"
mv "numdiff_u_"$first"-"$sec".txt" /c/Users/admin/Desktop/NUM/$2/"$first"-"$sec"
mv "lines_added_in"$first"-"$sec".txt" /c/Users/admin/Desktop/ADDED/$2/"$first"-"$sec"
cd /c/Users/admin/Desktop/DIVISION
mkdir -p $2
cd $2
mkdir -p "$first"-"$sec"
cd /c/Users/admin/Desktop
mv division_$first-$sec.txt  /c/Users/admin/Desktop/DIVISION/$2/"$first"-"$sec"




cd /c/Users/admin/Desktop
rm check_"$sec"-"$first".txt
rm check_"$first"-"$sec".txt
rm same2_"$first"-"$sec".txt
```

Εικόνα 22-Εκτέλεση 12 του Κώδικα

- Χρήση της εντολής mkdir για δημιουργία φακέλου
- Χρήση της εντολής rm με παράμετρο -rf για διαγραφή ολόκληρου φακέλου
- Χρήση της εντολής mv για μεταφορά των αρχείων στους κατάλληλους φακέλους και υποφακέλους
- Χρήση της εντολής rm χωρίς παράμετρο για διαγραφή αρχείων

Βιβλιογραφία

[1] Constantinou, E., &Stamelos, I. (2015). Architectural stability and evolution measurement for software reuse. Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC

[2] Jalender, B., A. Govardhan, and P. Premchand. "Breaking the boundaries for software component reuse technology." International Journal of Computer Applications 13.6 (2011): 37-41

[3] Fazal-e-Amin, Ahmad Kamil Mahmood, and Alan Oxley. "A Mixed Method Study to Identify Factors Affecting Software Reusability in Reuse Intensive Development." Computer and Information Sciences Department (2011)..

[4] Jalender, B., A. Govardhan, and P. Premchand. "Designing code level reusable software components." International Journal of Software Engineering&Applications 3.1 (2012): 219..

[5] Constantinou, Eleni, and IoannisStamelos. "Identifying evolution patterns: a metrics-based approach for external library reuse." *Software: Practice and Experience* 47.7 (2017): 1027-1039..

[6] Honglei, Tu, Sun Wei, and Zhang Yanan. "The research on software metrics and software complexity metrics." *2009 International Forum on Computer Science-Technology and Applications*. Vol. 1. IEEE, 2009.

[7] N. Schneidewind, "IEEE Standard For A Software Quality Metrics Methodology Revision And Reaffirmation," in Software Engineering Standards, International Symposium on, Walnut Creek, CA, 1997 pp. 278.

[8] Bartlett, James. "An introduction to JASP: A free and user-friendly statistics package." Recuperado de https://osf. io/p2hzg (2017).

[9] G. Xie, J. Chen and I. Neamtiu, "Towards a better understanding of software evolution: An empirical study on open source software," *2009 IEEE International Conference on Software Maintenance*, Edmonton, AB, Canada, 2009, pp. 51-60

[10] Amanatidis, T., &Chatzigeorgiou, A. (2016). Studying the evolution of PHP web applications. Information and Software Technology, 72, p.65

[11] Kaur, Taranjeet, Nisha Ratti, and Parminder Kaur. "Applicability of Lehman laws on open source evolution: a case study." *International Journal of Computer Applications* 93.18 (2014): 0975-8887.

[12] Neamtiu, Iulian, GuowuXie, and Jianbo Chen. "Towards a better understanding of software evolution: an empirical study on open-source software." *Journal of Software: Evolution and Process* 25.3 (2013): 193-218.

[13] Businge, John, Alexander Serebrenik, and Mark Van Den Brand. "An empirical study of the evolution of Eclipse third-party plug-ins." *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*. 2010.

[14] Israeli, Ayelet, and Dror G. Feitelson. "The Linux kernel as a case study in software evolution." *Journal of Systems and Software* 83.3 (2010): 485-501.

[15] Mens, Tom, Juan Fernández-Ramil, and Sylvain Degrandsart. "The evolution of Eclipse." *2008 IEEE International Conference on Software Maintenance*. IEEE, 2008.

[16] Robles, Gregorio, et al. "Evolution and growth in large libre software projects." *Eighth International Workshop on Principles of Software Evolution (IWPSE'05)*. IEEE, 2005.

[17] Godfrey, Michael, and Qiang Tu. "Growth, evolution, and structural change in open source software." *Proceedings of the 4th international workshop on principles of software evolution*. 2001.

[18] Lehman, Meir M. "Programs, cities, students—limits to growth?." *Programming Methodology*. Springer, New York, NY, 1978. 42-69.

[19] Pano, Amantia, Daniel Graziotin, and PekkaAbrahamsson. "Factors and actors leading to the adoption of a JavaScript framework." *Empirical Software Engineering* 23.6 (2018): 3503-3534.

[20] A. Terzi, S. Bibi and P. Sarigiannidis, "Reuse Opportunities in JavaScript applications," *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Palermo, Italy, 2021, pp. 387-391, doi: 10.1109/SEAA53835.2021.00057.

[21] Chatzimparmpas, A., Bibi, S., Zozas, I., &Kerren, A. (2019, May). Analyzing the Evolution of Javascript Applications. In *ENASE* (pp. 359-366)..

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

βλ.	βλέπε
κλπ.	Και λοιπά
κ.α.	και άλλα
API	Application Programming Interface
Json	JavaScript Object Notation
.txt	text only

Απόδοση Ξενόγλωσσων Όρων

Ξενόγλωσση	Απόδοση
Client side	Πλευρά των Πελατών
Front-end	Προσκήνιο
Back-end	Παρασκήνιο
Bugs	Σφάλματα
Cloud-based	Υπολογιστικό Νέφος
Branching	Διακλάδωση
Merging	Συγχώνευση
Most	Περισσότερο
Forks	Διακλάδωση
Package	Πακέτο
Branch	Κλάδος
Releases	Εκδόσεις
Years	Χρόνια
Days	Μέρες
Project	Έργο
Manager	Διευθυντής
Interface	Διεπιφάνεια