



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

TESI DI LAUREA MAGISTRALE IN INFORMATICA
CURRICULUM SOFTWARE ENGINEERING AND IT MANAGEMENT

Esiste una Relazione tra i Linguaggi di Programmazione e l'Insorgere di Community Smells? Uno Studio Empirico

RELATRICE

Prof.ssa **Filomena Ferrucci**

CO-RELATORI

Prof. **Fabio Palomba**

Dr. **Stefano Lambiase**

CANDIDATA

Alice Vidoni

Matricola: 0522500963

Anno Accademico 2021-2022

Sbagliare è umano, ma per incasinare davvero le cose serve un computer.

Abstract

Lo sviluppo di software è un'attività complessa composta da fattori tecnici e sociali. Tali aspetti, soprattutto quelli sociali, diventano ancora più centrali nel mondo dei software open-source, in cui comunicazione e collaborazione sono più complessi rispetto al mondo privato. Tale complessità è dovuta principalmente alla natura distribuita dei team open-source e alla loro struttura poco gerarchica vista la natura "libera" del software sviluppato. Per tali motivi, non è raro che in team open-source emergano pattern sociali che è stato dimostrato possono portare all'insorgere di problemi con conseguenze catastrofiche sia sul prodotto che sul processo. Per rappresentare gli aspetti di cui sopra, la letteratura ha definito i cosiddetti "*Community Smells*", un insieme di pattern sociali, spesso spiacevoli, che possono portare all'insorgere del fenomeno noto come *social debt*. Di fatto, in letteratura sono stati svolti diversi studi su come dei fattori sociali (cultura, lingua, posizione geografica, ecc.) influenzino l'insorgere di Community Smells, ma poco è stato fatto su un eventuale correlazione tra fattori tecnici e la presenza di Community Smells: nel dettaglio, è stata dimostrata l'influenza dei Community Smells sui fattori tecnici ma non il contrario. Pertanto, nel presente lavoro di tesi si è cercato di analizzare e illustrare se i fattori tecnici influenzino l'emergere di Community Smells. In particolare, è stato deciso di considerare come fattore tecnico il "*Linguaggio di Programmazione*", ossia il linguaggio di programmazione maggiormente presente in un codice sorgente. Tale scelta è stata presa tenendo in considerazione precedenti studi che hanno rivelato esserci un'influenza di Community Smells sul codice sorgente scritto dagli sviluppatori. Al fine di raggiungere l'obiettivo, è stato eseguito uno studio empirico riguardo delle caratteristiche di repository open-source di GitHub così da determinare i Community Smells presenti in esse, e successivamente è stato costruito un modello di regressione lineare multipla che mette in relazione il linguaggio di programmazione con l'emergere di dieci tipi di Community Smells, ovvero *Organizational Silo Effect*, *Black-cloud Effect*, *Prima-donnas Effect*, *Sharing Villainy*, *Organizational Skirmish*, *Solution Defiance*, *Radio Silence*, *Truck Factor Smell*, *Unhealthy Interaction* e *Toxic Communication*. I risultati ottenuti indicano che il linguaggio di programmazione utilizzato per l'implementazione di un prodotto software non influenza la presenza di Community Smells (ad eccezione del Community Smell Radio Silence) e, di conseguenza, non influenza le attività di collaborazione e comunicazione nelle comunità open-source.

Indice	ii
Elenco delle figure	v
Elenco delle tabelle	vii
1 Introduzione	1
1.1 Contesto Applicativo	1
1.2 Motivazioni e Obiettivi	2
1.3 Risultati Ottenuti	4
1.4 Struttura della tesi	5
2 Stato dell'arte	7
2.1 Aspetti umani dell'Ingegneria del Software	7
2.1.1 Social Debt	8
2.1.2 Community Smells	8
2.1.3 Metriche Socio-Tecniche	10
2.2 Influenza di linguaggi e paradigmi di programmazione	13
2.3 Tool CSDetector [18]	15
3 Metodologia di Ricerca	16
3.1 Ipotesi e Domande di Ricerca	16
3.2 Contesto di Studio	17
3.3 RQ₁ - Linguaggi di Programmazione su GITHUB	19

3.4	RQ₂ - Contributors in Repository GITHUB	20
3.5	RQ₃ - Esecuzione del tool CSDETECTOR	21
3.6	RQ₄ - Modelli di Regressione Lineare	26
3.6.1	Variabili Indipendenti	27
3.6.2	Variabile di Risposta	27
3.6.3	Variabili di Controllo	27
3.6.4	Costruzione del Modello Statistico	28
4	Analisi dei Risultati	30
4.1	Dataset: Statistiche di Sintesi e Informazioni Generali - RQ1	30
4.2	Dataset: Statistiche di Sintesi e Informazioni Generali - RQ2	32
4.3	Dataset: Statistiche di Sintesi e Informazioni Generali - RQ3	33
4.4	Modelli di Regressione - RQ4	36
4.4.1	Modello di Regressione per Community Smell: OSE	38
4.4.2	Modello di Regressione per Community Smell: BCE	38
4.4.3	Modello di Regressione per Community Smell: PDE	39
4.4.4	Modello di Regressione per Community Smell: SV	40
4.4.5	Modello di Regressione per Community Smell: OS	41
4.4.6	Modello di Regressione per Community Smell: SD	42
4.4.7	Modello di Regressione per Community Smell: RS	43
4.4.8	Modello di Regressione per Community Smell: TFS	44
4.4.9	Modello di Regressione per Community Smell: UI	45
4.4.10	Modello di Regressione per Community Smell: TC	45
5	Osservazioni sulla Ricerca e sui Risultati	49
5.1	Osservazioni	49
5.1.1	Esecuzione del Tool CSDETECTOR su repository (RQ₃)	50
5.1.2	Modello Statistico (RQ₄)	50
5.2	Threats to Validity	51
5.2.1	Threats to Construct Validity	51
5.2.2	Threats to Internal Validity	52
5.2.3	Threats to External Validity	52
6	Conclusioni	53
6.1	Conclusioni	53

6.2	Lavori Futuri	54
7	Ringraziamenti	56
	Bibliografia	58

Elenco delle figure

2.1	Panoramica dell'approccio utilizzato in CSDetector	15
3.1	Panoramica del processo di ricerca	18
3.2	Struttura set di dati di GH Archive.	19
3.3	Query eseguita su Google BigQuery per ottenere le repository per linguaggio.	21
3.4	Query eseguita su Google BigQuery per ottenere i contributors di ogni repository.	22
3.5	Suddivisione dataset per applicare Regressione Lineare.	29
4.1	Linguaggi di Programmazione maggiormente utilizzati	31
4.2	Linguaggi di Programmazione maggiormente utilizzati	31
4.3	Media numero di contributors per ogni linguaggio.	33
4.4	Panoramica della gestione dei risultati del Tool CSDetector	34
4.5	Percentuale di presenza di uno Smell nei Linguaggi di Programmazione	35
4.6	Risultati ottenuti dal modello per il Community Smell OSE	39
4.7	Risultati ottenuti dal modello per il Community Smell BCE	40
4.8	Risultati ottenuti dal modello per il Community Smell PDE	41
4.9	Risultati ottenuti dal modello per il Community Smell SV	42
4.10	Risultati ottenuti dal modello per il Community Smell OS	43
4.11	Risultati ottenuti dal modello per il Community Smell SD	44
4.12	Risultati ottenuti dal modello per il Community Smell RS	45
4.13	Risultati ottenuti dal modello per il Community Smell TFS	46
4.14	Risultati ottenuti dal modello per il Community Smell UI	47

4.15 Risultati ottenuti dal modello per il Community Smell TC	48
---	----

Elenco delle tabelle

1.1	Panoramica sui community smells analizzati [18]	4
2.1	Panoramica dei Community Smells più comuni	9
2.2	Fattori di qualità sociale e socio-tecnica [16].	12
2.3	Categorie di bug e loro distribuzione nel set di dati [19]	14
3.1	Metriche Socio-Tecniche restituite da CSDETECTOR	24
4.1	Linguaggi di programmazione e le loro caratteristiche.	32
4.2	Presenza di Community Smells per Linguaggio	36
4.3	Panoramica sui community smells analizzati [18]	37

1.1 Contesto Applicativo

Al giorno d'oggi, il software è pervasivo nella vita delle persone. Per questo motivo, il suo sviluppo e la sua progettazione sono attività essenziali in grado di influenzare vari aspetti, primo fra tutti la vita umana. Pertanto, è importante notare che fino al 69% dei progetti software fallisce [1, 2, 3, 4]. Diversi lavori di ricerca hanno riportato alcune conseguenze di tali fallimenti: ad esempio, il 17% porta al fallimento dell'azienda. Il progetto *Virtual Case File Project* (per il Federal Bureau of Investigation degli Stati Uniti) è il fallimento meglio documentato che, secondo Frieden [5], ha portato alla perdita di cinque anni di sviluppo e di 170 milioni di dollari.

Per questi motivi, conoscere meglio i fattori fondamentali che influenzano l'esito di un progetto è obbligatorio per migliorare la percentuale di progetti di successo. Lo stato dell'arte ha identificato diverse cause tecniche e sociali legate a due campi specifici: *Ingegneria del software* (SE) e *Project Management* (PM) [1, 4, 3, 6]. Tra questi fattori spiccano: (1) sottostima dei costi del progetto, (2) pessima gestione del rischio, (3) mancata ricompensa del personale per lo straordinario e (4) incapacità di comunicare e agire come un team.

In sintesi, è possibile affermare che l'Ingegneria del Software e il Project Management sono due dei fattori più critici che influenzano il risultato del progetto. In questi campi, due problemi emergono dalla letteratura moderna:

- **Aspetti Sociali** - Anche se il Project Management è un'attività influenzata principal-

mente da aspetti umani e sociali [7, 8, 9], la maggior parte della ricerca in questo campo riguarda processi tecnici e metriche.

- **Ingegneria del Software Globale** - Al giorno d'oggi, un elevato numero di progetti è open-source (cioè rilasciato con una licenza in cui il titolare del copyright concede agli utenti i diritti di utilizzare, studiare, modificare e distribuire il software e il relativo codice sorgente a chiunque e per qualsiasi scopo), e ciò comporta una scarsa o quasi nulla collaborazione e comunicazione tra gli sviluppatori e/o comunità [10, 11].

1.2 Motivazioni e Obiettivi

Dato il contesto descritto al punto precedente, l'obiettivo principale di questa tesi di ricerca combina entrambi i problemi (Aspetti Sociali e Ingegneria del Software Globale) con l'intento di analizzare e illustrare come l'utilizzo di un linguaggio di programmazione influenzi la collaborazione e la comunicazione tra gli sviluppatori, favorendo la presenza di Community Smells.

Lo sviluppo di software è un'attività complessa composta da fattori tecnici e sociali. Ad oggi, la maggior parte delle ricerche in letteratura si sono concentrate su aspetti tecnici (per esempio le metriche del software), mentre gli aspetti sociali e i fattori che li influenzano non sono stati molto approfonditi. Tuttavia, l'attuale stato dell'arte presenta alcune interessanti ricerche riguardo i fattori che influenzano l'insorgere di problemi sociali, e quindi Community Smells. Catolino et al. [12] hanno condotto uno studio empirico indagando la correlazione tra la diversità di genere e l'emergere di Community Smells. Lambiase et al. [13], invece, hanno condotto uno studio empirico indagando la correlazione tra la dispersione geografica e culturale e l'emergere di Community Smells. Entrambi gli studi si sono basati sui Community Smells: Organizational Silo, Black Cloud, Lone Wolf e Radio Silence. Ulteriore interessante ricerca è stata condotta da Palomba et al. [14], che hanno eseguito un'indagine empirica per studiare una relazione tra Community Smells e aspetti legati al codice sorgente, quali i Code Smells, ed eventuali possibili impatti.

Obiettivo Principale

Analizzare e illustrare, attraverso l'uso di una strategia di ricerca quantitativa, se il linguaggio di programmazione utilizzato influenzi l'emergere di problemi di collaborazione e comunicazione, rappresentati attraverso i Community Smells, all'interno dei team di sviluppo software open-source.

Per il presente studio, sono stati individuati i linguaggi di programmazione maggiormente utilizzati nei progetti caricati su GITHUB, e presi in considerazione i primi 10 che presentano caratteristiche differenti tra di loro. Successivamente, è stata eseguita un'analisi su repository in cui il linguaggio di programmazione in questione predilige.

Sono stati individuati i problemi sociali delle repository selezionate attraverso la presenza e l'analisi di *community smells*, ovvero modelli non ottimali nella struttura organizzativa e sociale di una comunità di sviluppo software che sono precursori di eventi socio-tecnici allarmanti e imprevisti [15, 14, 16, 17]. Per aiutare meglio la comprensione di tali "smells", riportiamo, a titolo di esempio, due classi: *Black-cloud Effect* e *Organizational Silo Effect*. Si ha un *Black-cloud Effect* quando vi è un sovraccarico di informazioni dovuto alla mancanza di comunicazioni strutturate a causa delle limitate opportunità di condivisione delle conoscenze (ad esempio, collaborazioni, discussioni, stand-up quotidiani, ecc.), nonché alla mancanza di membri esperti nel progetto in grado di coprire l'esperienza o il divario di conoscenza di una comunità. D'altro lato, abbiamo un *Organizational Silo Effect* quando vi è la presenza di sottogruppi isolati e mancanza di comunicazione e collaborazione tra gli sviluppatori della comunità. Di conseguenza, questo smell causa un costo imprevisto aggiuntivo per un progetto a causa dello spreco di risorse (ad esempio il tempo), nonché della duplicazione del codice.

Per raggiungere questo obiettivo, sono stati condotti degli studi quantitativi. In primo luogo, sono state effettuate diverse analisi per studiare i linguaggi di programmazione maggiormente utilizzati nelle comunità open-source, in particolare su GITHUB (una piattaforma per lo sviluppo collaborativo di software e il controllo delle versioni che utilizza GIT). Dopodiché, è stata condotta un'analisi sugli aspetti sociali presenti nelle repository selezionate, attraverso l'individuazione di Community Smells e metriche socio-tecniche per ognuna di esse. Come studio quantitativo, è stata utilizzata la *regressione lineare multipla* per costruire un modello che mette in relazione il linguaggio di programmazione utilizzato con la presenza di alcuni Community Smells, in particolare *Organizational Silo Effect*, *Black-cloud Effect*, *Primadonnas Effect*, *Sharing Villainy*, *Organizational Skirmish*, *Solution Defiance*, *Radio Silence*, *Truck*

Tabella 1.1: Panoramica sui community smells analizzati [18]

Community Smell	Definizione	Conseguenza
Organizational Silo Effect	Si riferisce alla presenza di sottogruppi isolati e alla mancanza di comunicazione e collaborazione tra gli sviluppatori della comunità.	Causa un costo imprevisto aggiuntivo per un progetto a causa dello spreco di risorse (ad esempio, tempo), nonché della duplicazione del codice.
Black Cloud Effect	Eccessivo sovraccarico di informazioni dovuto alla mancanza di una comunicazione strutturata o di una governance della cooperazione.	Mancanza di membri esperti nel progetto in grado di coprire l'esperienza o il divario di conoscenza di una comunità.
Radio Silence	Si verifica un'elevata formalità di procedure regolari a causa dell'inefficiente organizzazione strutturale di una comunità.	Causa un enorme ritardo inaspettato nel processo decisionale dovuto alle necessarie azioni formali necessarie.
Prima Donna	Ripetuto comportamento condiscendente, superiorità, disaccordo costante, mancanza di collaborazione da parte di uno o pochi membri.	Collaborazione strutturata in modo inefficiente all'interno di una comunità.
Sharing Villainy	Mancanza di attività di scambio di informazioni di qualità.	i membri della comunità condividono conoscenze essenziali come informazioni obsolete, errate e non confermate.
Organizational Skirmish	Causato da un disallineamento tra diversi livelli di competenza e canali di comunicazione tra unità di sviluppo o individui coinvolti nel progetto.	Calo della produttività e influisce sulla tempistica e sui costi del progetto.
Solution Defiance	La comunità di sviluppo presenta diversi livelli di background culturale ed esperienziale e queste variazioni portano alla divisione della comunità in sottogruppi simili con opinioni completamente contrastanti sulle decisioni tecniche o socio-tecniche da prendere.	Ritardi imprevisti del progetto e comportamenti non collaborativi tra gli sviluppatori.
Truck Factor Smell	La maggior parte delle informazioni e delle conoscenze sul progetto sono concentrate in uno o pochi sviluppatori.	Significativa perdita di conoscenza a causa del turnover degli sviluppatori.
Unhealthy Interaction	Le discussioni tra gli sviluppatori sono lente, leggere, brevi e/o contengono conversazioni scadenti.	Bassa partecipazione degli sviluppatori alle discussioni del progetto (ad esempio, pull requests, problemi, ecc.) con lunghi ritardi tra le comunicazioni dei messaggi.
Toxic Communication	Le comunicazioni tra gli sviluppatori sono soggette a conversazioni tossiche e sentimenti negativi contenenti opinioni spiacevoli, di rabbia o addirittura contrastanti su vari argomenti di cui le persone discutono.	Gli sviluppatori possono avere interazioni interpersonali negative con i loro coetanei, che possono portare a frustrazione e stress. Queste interazioni negative possono alla fine portare gli sviluppatori ad abbandonare i progetti.

Factor Smell, Unhealthy Interaction e Toxic Communication.

La Tabella 1.1 riporta gli smells analizzati, con la loro definizione e le eventuali conseguenze che si ripercuotono sulla qualità del codice prodotto e nel team di sviluppo.

Lo stato dell'arte ha già dimostrato che il linguaggio di programmazione utilizzato influisce sulla qualità del codice prodotto[19], come dettagliato nel capitolo 2 di questa tesi.

1.3 Risultati Ottenuti

In questa tesi di ricerca, è stato esteso lo stato dell'arte fornendo approfondimenti per quanto riguarda l'influenza del linguaggio di programmazione utilizzato per lo sviluppo, sulla presenza di Community Smells, e quindi su eventuali problemi di comunicazione e collaborazione in una comunità. In particolare, sono stati forniti i seguenti contributi:

- Analisi dei Linguaggi e Costruzione del Dataset** - È stata condotta un'analisi per l'individuazione dei linguaggi maggiormente utilizzati in progetti open-source presenti su GITHUB e, successivamente, sono stati individuati dieci linguaggi con caratteristiche differenti tra loro. A partire dall'elenco dei linguaggi individuati è stato costruito un dataset contenente le informazioni su 200 repository open-source in cui nel codice sorgente predilige uno dei linguaggi di programmazione individuati nella fase precedente.
- Esecuzione del Tool CSDetector e Analisi dei Risultati** - È stato implementato uno script in Python per permettere l'esecuzione del tool CSDetector su tutte le repository del dataset. Successivamente sono stati analizzati i Community Smells e le metriche

socio-tecniche restituite in output dal tool, con la conseguente individuazione di quelle più utili ai fini dello studio statistico.

3. **Modello Statistico** - È stato costruito un modello di regressione lineare statistica per studiare l'influenza del linguaggio di programmazione principalmente utilizzato nel codice sorgente sulla presenza di Community Smells nelle comunità di software, per formalizzare il legame tra il linguaggio di programmazione primario nello sviluppo e le attività di comunicazione e collaborazione.
4. **Repository Online** - È stata messa a disposizione una repository [20] GITHUB per rendere i dati e gli script pubblicamente disponibili, e fornire materiale che altri possano utilizzare per comprendere i nostri risultati o svolgere nuovi studi.

I risultati dello studio hanno rivelato che non sembra esistere un'influenza tra il linguaggio di programmazione principalmente utilizzato nel codice sorgente e l'emergere di problemi di comunicazione e collaborazione all'interno della specifica comunità di sviluppo software. L'unica eccezione si è verificata per il Community Smell *Radio Silence*, che indica un'elevata formalità di procedure regolari a causa dell'inefficiente organizzazione strutturale di una comunità di sviluppo.

1.4 Struttura della tesi

Tutti i temi sopra menzionati sono stati proposti in modo approfondito nei cinque capitoli di cui si compone il presente lavoro di tesi. Di seguito una sintesi dei capitoli:

- **Capitolo 2: Stato dell'arte**, illustra le opere e le scoperte della letteratura attualmente disponibili nel campo degli aspetti umani e dello sviluppo del software;
- **Capitolo 3: Metodologia di ricerca**, che illustra le domande di ricerca e le strategie di ricerca utilizzate per raggiungere l'obiettivo principale;
- **Capitolo 4: Analisi dei risultati**, che illustra i risultati raggiunti per ciascuna domanda di ricerca e le conclusioni relative all'obiettivo principale;
- **Capitolo 5: Osservazioni sulla Ricerca e sui Risultati**, che illustra le osservazioni sulla ricerca, sui risultati ottenuti, le threats to validity dello studio e il modo in cui queste ultime sono state attenuate;

- **Capitolo 6: Conclusioni**, che fornisce una sintesi della ricerca e delinea alcuni possibili lavori futuri.

Questo capitolo illustra lo stato dell'arte e il lavoro presente in letteratura sugli aspetti di ricerca interessati dal nostro studio.

2.1 Aspetti umani dell'Ingegneria del Software

La comunità di ricerca del Software Engineering (SE), fin dai primi studi svolti, si è concentrata principalmente sugli aspetti tecnici del software [21, 22, 23, 24, 25]. Tale attività di ricerca è giunta al termine con la definizione di “*technical debt*” (ossia, il costo aggiuntivo causato da pratiche di programmazione che portano a soluzioni di implementazione scadenti che riducono la qualità del codice sorgente) data da Cunningham [26].

Tuttavia, la SE è, per sua natura, un'attività “sociale” che coinvolge organizzazioni, sviluppatori e stakeholders, responsabili di portare alla definizione di un prodotto che soddisfi i requisiti attesi [27]. Infatti, già nel 1975, Fred Brooks, nel suo libro ‘*The Mythical Man-Month: Essays on Software Engineering*’ [8], citava al fatto che la composizione dei team e le loro tecniche di management potevano influenzare il successo dei progetti software legati allo sviluppo. Nella sezione seguente si riportano i principali contributi della letteratura moderna al problema degli aspetti sociali nello sviluppo e nella gestione del software.

2.1.1 Social Debt

Per quanto concerne la prospettiva sociale e umana in SE, la maggior parte della letteratura passata si è concentrata sul *social debt* (debito sociale), cioè i costi impreveduti di un progetto legati ad una comunità di sviluppo non ottimale [28, 29]. Per esempio, Tamburri et al. [29] hanno illustrato il social debt a confronto con il technical debt, e hanno discusso scenari comuni di vita reale che mostrano team di sviluppo “sub-ottimali”. Inoltre, Tamburri et al. [28] hanno condotto una ricerca qualitativa esplorativa in una grande azienda di sviluppo software con l’obiettivo di studiare le cause intorno al social debt in contesti pratici. Quest’ultimo lavoro ha definito un quadro contenente le cause comuni del social debt e alcune buone pratiche adottate per ripagare parte del debito accumulato. Entrambe le ricerche hanno rivelato che il social debt è fortemente correlato al technical debt, e che entrambi gli aspetti dovrebbero essere considerati insieme durante il processo di gestione del software.

2.1.2 Community Smells

Come evoluzione della ricerca sul social debt, Tamburri et al. hanno definito i *Community Smells*, cioè un insieme di caratteristiche socio-tecniche (per esempio, l’alta formalità) e di patterns (per esempio, comportamenti indulgenti ricorrenti, o abbandono del team in preda alla rabbia), che possono portare all’emergere del social debt. La Tabella 2.1 riporta i tipici Community Smells identificati dalla letteratura [15, 28, 17].

Negli ultimi anni i Community Smells hanno iniziato a ricevere particolare attenzione. Una delle motivazioni risiede nello sviluppo di strumenti in grado di rilevare tali smells utilizzando varie strategie. Tamburri, Palomba e Kazman [30] hanno proposto un tool CODEFACE4SMELLS, una versione avanzata CODEFACE di Joblin et al. [31], capace di rilevare i Community Smells. Inoltre, Tamburri, Palomba e Kazman [30] hanno rilevato le capacità di rilevazione di CODEFACE4SMELLS utilizzando un sondaggio ed eseguendo una valutazione empirica su un insieme di 60 progetti. Tale attività ha confermato la capacità dello strumento nel riuscire a rilevare correttamente i Community Smells. Oltre questo studio, Palomba et al. [14] hanno usato CODEFACE4SMELLS per studiare l’impatto di tali smells sui *code smells*.¹ Tale studio ha dimostrato che i Community Smells rappresentano fattori chiave che impediscono agli sviluppatori di eseguire attività di refactoring.

¹*code smells* sono delle “scelte implementative sbagliate, applicate dagli sviluppatori durante l’evoluzione del software, che spesso portano a difetti critici o al fallimento di quest’ultimo” [14] [32].

Tabella 2.1: Panoramica dei Community Smells più comuni

Community Smell	Definizione
Organizational Silo	Aree isolate della comunità di sviluppo che non comunicano, se non attraverso uno o due dei loro rispettivi membri.
Black Cloud	Eccessivo sovraccarico di informazioni dovuto alla mancanza di una comunicazione strutturata o di una governance della cooperazione.
Lone Wolf	Sviluppatore sfiduciato che apporta modifiche al codice sorgente senza considerare le opinioni dei suoi colleghi.
Radio Silence	Un membro si interpone in ogni interazione tra le sottocomunità.
Prima Donna	Comportamento accondiscendente ripetuto, superiorità, disaccordo costante, non collaborazione da parte di uno o pochi membri.
Priggish Members	Esigere dagli altri un'inutile e precisa conformità o un'esagerata correttezza, soprattutto in modo moraleggiante o irritante.
Code Red	Questo smell identifica un'area di codice così complessa, densa e dipendente da 1-2 manutentori che sono gli unici a poter fare un refactoring.
Unlearning	Un nuovo progresso tecnologico o organizzativo o una best practice che diventa impraticabile se condivisa con i membri più anziani.
Disengagement	Pensare che il prodotto sia abbastanza maturo e inviarlo alle operazioni anche se potrebbe non essere pronto.
Cognitive Distance	Gli sviluppatori percepiscono la distanza a livello fisico, tecnico, sociale e culturale rispetto a coetanei con notevoli differenze di background.

Un trend più recente è rappresentato dalla definizione di meccanismi di modellazione in grado di descrivere la struttura futura di un team e di avvisare i project manager della presenza del social debt [18, 33, 34]. Per contribuire a questo studio, Almarimi et al. [33] hanno costruito un modello di apprendimento multi-label basato su algoritmi genetici per rilevare otto tipi comuni di Community Smells. La valutazione del modello ha coinvolto 103 progetti open-source e 407 istanze di community smells, e ha dato come risultato indici di performance migliori rispetto ad altre soluzioni (F-measure dell'89%).

I ricercatori hanno anche studiato i fattori sociali significativi che influenzano i Community Smells e il modo in cui i manager li gestiscono. Catolino et al. [12] hanno condotto uno studio empirico indagando la correlazione tra la diversità di genere e l'emergere di quattro tipi di Community Smells (Organizational Silo, Black Cloud, Lone Wolf e Radio Silence). Tra i risultati ottenuti, uno dei più interessanti consiste nell'influenza delle donne sull'emergere di Community Smells correlati alla qualità della comunicazione (ad esempio, l'effetto *Radio Silence*). Nello specifico, per le strategie di refactoring Catolino et al. [35],

hanno studiato anche come gli sviluppatori rimuovono i Community Smells. Attraverso un sondaggio condotto con 76 esperti, gli autori hanno ottenuto una serie di operazioni di refactoring, generalmente applicate dai professionisti, per rimuovere quattro tipi di smells. Dalle strategie identificate emergono il mentoring, il monitoraggio, la creazione di un piano di comunicazione, la conduzione di esercizi di coesione e la ristrutturazione del team.

Lambiase et al. [13], invece, hanno condotto uno studio empirico indagando la correlazione tra la dispersione culturale e geografica del team di sviluppo e l'emergere di quattro tipi di Community Smells (Organizational Silo, Black Cloud, Lone Wolf e Radio Silence). I risultati ottenuti hanno indicato che i fattori culturali e geografici influenzano la collaborazione e la comunicazione all'interno delle comunità open-source, in misura tale da incitare o, cosa ancora più interessante, da mitigare, in alcuni casi, i Community Smells, ad esempio *Lone Wolf*, nei team di sviluppo. Emerge che i manager possono utilizzare questi risultati come supporto per la costruzione della struttura dei team.

Per quanto riguarda l'impatto che tali Community Smells hanno sull'intero campo dello sviluppo del software, sono stati condotti vari studi. Martini e Bosch [36] hanno studiato l'impatto dei Community Smells sul debito dell'architettura, mentre Tamburri et al. [37] sui tipi di struttura organizzativa.

Community Smells

I *Community Smells* sono un insieme di caratteristiche socio-tecniche (ad esempio, alta formalità) e modelli (ad esempio, comportamento indulgente ricorrente, o abbandono del team in preda alla rabbia) in un team di sviluppo software, che sono precursori di eventi socio-tecnici allarmanti e imprevisti come il social debt [15, 17].

2.1.3 Metriche Socio-Tecniche

Lo sviluppo di un progetto software coinvolge due elementi fondamentali: una componente tecnica e una componente sociale. La componente tecnica può essere considerata come la combinazione di (1) proprietà tecniche del prodotto da sviluppare, (2) i processi di sviluppo e (3) le tecnologie utilizzate durante il ciclo di sviluppo del prodotto. La componente sociale consiste (1) nell'organizzazione e negli individui coinvolti nel processo di sviluppo, (2) nei loro atteggiamenti, (3) nei loro comportamenti, e (4) nelle relazioni tra loro. Indubbiamente, un progetto di sviluppo di un prodotto può essere visto come un *sistema socio-tecnico* in cui le due componenti discusse sopra devono essere allineate e gestite per garantire un risultato di

successo. Inoltre, non dovrebbe sorprendere che le due componenti possano influenzarsi a vicenda e come la gestione di questa relazione richieda uno sforzo considerevole.

Per rappresentare e rendere operativo un tale comportamento socio-tecnico lo stato dell'arte ha proposto varie metriche. Tamburri, Palomba e Kazman [16] hanno condotto una *revisione sistematica della letteratura* (SLR) che ha indagato i fattori sociali e socio-tecnici per l'ingegneria del software [38, 29, 39, 40]. La Tabella 2.2 riporta questi fattori dividendoli per categoria e fornendo una breve descrizione. Inoltre, Magnoni et al. [41] hanno proposto un *Framework di qualità socio-tecnica*, composto dai fattori riportati nella Tabella 2.2, per studiare quali di essi influenzano principalmente l'emergere del social debt.

Tabella 2.2: Fattori di qualità sociale e socio-tecnica [16].

Categoria	Metrica	Descrizione
Metriche Developer Social Network	devs	Numero di sviluppatori presenti nel Developers Social Network globale
	ml.only.devs	Numero di sviluppatori presenti solo nella comunicazione nel Developers Social Network
	code.only.devs	Numero di sviluppatori presenti solo nella collaborazione nel Developers Social Network
	ml.code.devs	Numero di sviluppatori presenti sia nella comunicazione che nella collaborazione nel DSNs
	perc.ml.only.devs	Percentuale degli sviluppatori presenti solo nella comunicazione nel Developers Social Network
	perc.code.only.devs	Percentuale degli sviluppatori presenti solo nella collaborazione nel Developers Social Network
	perc.ml.code.devs	Percentuale degli sviluppatori presenti sia nella comunicazione che nella collaborazione nel DSNs
	sponsored.devs	Numero di sviluppatori sponsorizzati (95% dei loro commit sono fatti in orario di lavoro)
	ratio.sponsored	Rapporto tra sviluppatori sponsorizzati e sviluppatori presenti nella collaborazione nel DSN
Metriche Socio-Tecniche	st.congruence	Stima della Socio-Technical Congruence
	communicability	Stima della trasmissione delle informazioni (diffusione delle decisioni)
	num.tz	Numero di fusi orari coinvolti nello sviluppo del software
	ratio.smelly.devs	Rapporto di sviluppatori coinvolti in almeno un Community Smell
Metriche sui membri di team core	core.global.devs	Numero di sviluppatori core nel Developers Social Network globale
	core.mail.devs	Numero di sviluppatori core nella comunicazione nel Developers Social Network
	core.code.devs	Numero di sviluppatori core nella collaborazione nel Developers Social Network
	sponsored.core.devs	Numero di sviluppatori core sponsorizzati
	ratio.sponsored.core	Rapporto tra gli sviluppatori core sponsorizzati e gli sviluppatori core della collaborazione nel DSN
	global.truck	Rapporto di sviluppatori non-core nel Developers Social Network globale
	mail.truck	Rapporto di sviluppatori non-core della comunicazione nel Developers Social Network
	code.truck	Rapporto di sviluppatori non-core della collaborazione nel Developers Social Network
	mail.only.core.devs	Numero di sviluppatori core presenti solo nella comunicazione nel DSN
	code.only.core.devs	Numero di sviluppatori core presenti solo nella collaborazione nel DSN
	ml.code.core.devs	Numero di sviluppatori core presenti sia nella comunicazione sia nella collaborazione nel DSNs
	ratio.mail.only.core	Rapporto di sviluppatori core presenti solo nella comunicazione nel DSN
	ratio.code.only.core	Rapporto di sviluppatori core presenti solo nella collaborazione nel DSN
	ratio.ml.code.core	Rapporto di sviluppatori core presenti sia nella comunicazione che nella collaborazione nel DSNs
Turnover (fatturato)	global.turnover	Il turnover (fatturato) globale degli sviluppatori rispetto alla finestra temporale precedente
	code.turnover	Il turnover (fatturato) degli sviluppatori in collaborazione rispetto alla finestra temporale precedente
	core.global.turnover	Il turnover (fatturato) degli sviluppatori core rispetto alla finestra temporale precedente
	core.mail.turnover	Il turnover (fatturato) degli sviluppatori core nella comunicazione rispetto alla finestra temporale precedente
	core.code.turnover	Il turnover (fatturato) degli sviluppatori core nella collaborazione rispetto alla finestra temporale precedente
	ratio.smelly.quitters	Rapporto degli sviluppatori precedentemente coinvolti in Community Smells che hanno abbandonato il team
Metriche di Analisi Social Network	closeness.cent	La metrica del grado SNA del DSN (Developers Social Network) globale calcolata usando la vicinanza
	betweenness.cent	La metrica del grado SNA del DSN globale calcolata usando la centralità
	degree.cent	La metrica del grado SNA del DSN globale calcolata usando il grado
	global.mod	Metrica di modularità SNA del DSN globale
	mail.mod	Metrica di modularità SNA della comunicazione in DSN
	code.mod	Metrica di modularità SNA della collaborazione in DSN
	density	Metrica della densità SNA del DSN globale

Socio-Technical Congruence

Oltre agli studi sul social debt e sulle metriche socio-tecniche, sono state realizzate diverse analisi empiriche sulla cosiddetta *Socio-Technical Congruence* [42, 43, 44, 45]. L'idea principale della Socio-Technical Congruence si basa sulla *Legge di Conway*, che porta il nome di Melvin Conway, che diceva “*Qualsiasi organizzazione che progetta un sistema (definito a grandi linee) produrrà un progetto la cui struttura è una copia della struttura di comunicazione dell'organizzazione stessa.*” [46]. Sulla base di questa legge, la Socio-Technical Congruence (STC) può essere definita come una misura che valuta l'allineamento tra i requisiti di coordinamento (estratti dalle dipendenze tecniche tra i compiti) e le effettive attività di coordinamento eseguite dagli sviluppatori [42, 43].

Diversi lavori hanno dimostrato i vantaggi di ottenere un buon livello di STC, in particolare sulla produttività [42] e sulla qualità [47]. Inoltre, sono stati proposti alcuni strumenti che possono misurare la STC e gli aspetti relativi alla STC, come Ariadne [48] o Tesseract [49], progettato per visualizzare le relazioni di coordinamento e le lacune nei team di sviluppo software. Nel contesto della STC in Global Software Engineering, Portillo-Rodríguez et al. [45] hanno progettato un'architettura ad agenti per la coordinazione e la comunicazione in grado di misurare la STC nei team distribuiti. Tale proposta è stata convalidata attraverso un caso di studio eseguito presso Indra Software Labs con risultati positivi.

Più correlati al nostro studio, Palomba et al. [14] hanno condotto un'indagine empirica per affrontare la coesistenza e gli effetti tra i Community Smells e Code Smells. Palomba et al. [50] hanno intervistato vari sviluppatori per valutare preliminarmente se esiste una tale influenza. Tali studi hanno riportato che i Community Smells rappresentano fattori chiave che impediscono agli sviluppatori di eseguire attività di refactoring.

2.2 Influenza di linguaggi e paradigmi di programmazione

Elemento primario nello sviluppo di un software è il linguaggio di programmazione utilizzato, e quindi i paradigmi che quest'ultimo fornisce per la stesura del codice. Diversi studi hanno avuto come obiettivo quello di determinare se il linguaggio di programmazione utilizzato per la stesura del codice abbia una correlazione con i bugs all'interno di esso, e quindi, se il linguaggio di programmazione abbia un'influenza sulla qualità del codice prodotto.

In particolare, uno studio di Ray et al. [19] ha raggiunto conclusioni molto interessanti. Grazie all'utilizzo di GitHub Archive e GitHub Linguist, gli autori hanno selezionato i

Tabella 2.3: Categorie di bug e loro distribuzione nel set di dati [19]

Tipo di Bug		Descrizione del Bug	Parole/Frasi Chiave
Cause	Algorithm	errori algoritmici o logici	algorithm
	Concurrency	problemi relativi al multi-threading o al multi-processing	deadlock, race condition, synchronization error
	Memory	gestione errata della memoria	memory leak, null pointer, buffer overflow, heap overflow, null pointer, dangling pointer, double free, segmentation fault
	Programming	errori generici di programmazione	exception handling, error handling, type error, typo, compilation error, copy-paste error, refactoring, missing switch case, faulty initialization, default value
Impact	Security	funziona correttamente ma può essere sfruttato da hackers	by attackers buffer overflow, security, password, oauth, ssl
	Performance	funziona correttamente ma è lento nel ritornare una risposta	optimization problem, performance
	Failure	crash or hang	reboot, crash, hang, restart
Unknown		non fa parte delle sette categorie di cui sopra	

diciannove linguaggi di programmazione maggiormente utilizzati (escludendo Shell, CSS e altri linguaggi non ritenuti di uso generale) e, per ognuno di questi linguaggi, sono state selezionate cinquanta repository GitHub open-source con maggior numero di contributors e con alto gradimento. Su queste repository è stata svolta un’analisi del testo dei commit. Ciò ha consentito, oltre che l’individuazione di bugs tramite parole chiave, anche di determinare come questi siano stati risolti.

Per sostenere questa analisi i bugs sono stati classificati come riportato nella Tabella 2.3 [19] e i linguaggi di programmazione sono stati classificati in base alle loro caratteristiche (procedurali, funzionali, di scripting, tipizzazione forte, tipizzazione debole, statici, dinamici, con memoria gestita e con memoria non gestita).

Le conclusioni di questo studio hanno dimostrato che alcuni linguaggi di programmazione hanno una maggiore correlazione con bugs rispetto ad altri, mentre la correlazione tra bugs e classe di appartenenza di un linguaggio è meno evidente ma comunque significativa (per esempio i linguaggi funzionali hanno una correlazione minore con bugs rispetto a linguaggi procedurali o linguaggi di scripting). È stato dimostrato anche l’opposto, ossia che la correlazione non è solo tra linguaggio e bugs, ma anche tra classe di bug e linguaggio (ad esempio alcuni tipi di bug come l’errore di memoria o gli errori di concorrenza dipendono anche dalle primitive del linguaggio).

È importante precisare che la qualità di un prodotto software è influenzata da innumerevoli fattori (per esempio la dimensione del prodotto, il numero di sviluppatori coinvolti, ecc.), ma è stato fermamente dimostrato che il linguaggio di programmazione utilizzato è un importante contributo alla qualità del codice.

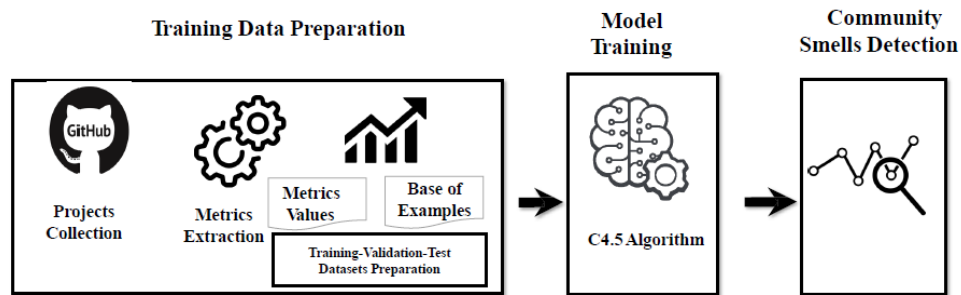


Figura 2.1: Panoramica dell'approccio utilizzato in CSDetector

2.3 Tool CSDetector [18]

Affinchè sia possibile individuare e scoprire tempestivamente l'esistenza di potenziali Community Smells in un progetto software, Almarimi, Ouni e Mkaouer [18] hanno implementato un tool denominato CSDetector. CSDetector utilizza un approccio di rilevamento basato sull'apprendimento automatico, che apprende da varie pratiche di sviluppo di community scorrette per fornire un supporto automatico nell'individuazione di tali Community Smells. In particolare, è stato utilizzato un approccio che apprende da un insieme di sintomi organizzativi e sociali che caratterizzano l'esistenza di potenziali istanze di Community Smells in un progetto software. Il modello di rilevamento è stato costruito utilizzando l'albero delle decisioni e adottando diversi algoritmi di apprendimento automatico quali C4.5, JRip, Random Forest, Nave Bayes, SMO e Lib-SVM. Tuttavia, il classificatore C4.5 è risultato essere l'algoritmo più accurato per l'identificazione dei Community Smells. Almarimi, Ouni e Mkaouer, per valutare le prestazioni e l'affidabilità del loro approccio, hanno condotto uno studio empirico su un benchmark di 74 progetti open-source di natura diversa (in termini di dimensioni, dominio di applicazione) presenti su GITHUB. I risultati ottenuti hanno mostrato che CSDetector raggiunge un'accuratezza del 96%, un'AUC (Area Under the Curve) di 0,94 e che il tool risulta essere migliore in termini di performance e di accuratezza rispetto ad altri tool presenti online. Nella Figura 2.1 viene mostrato l'approccio utilizzato.

Il tool CSDetector è in grado di rilevare 10 Community Smells, che sono nel dettaglio descritti nella Tabella 1.1, dato che risultano essere noti per i loro effetti negativi nella pratica, e che quindi più soggetti a contribuire al social debt e ad una scarsa qualità del codice sorgente.

Questo capitolo illustra le domande di ricerca e le strategie di ricerca utilizzate per raggiungere l'obiettivo principale.

L'obiettivo principale dello studio è quello di valutare se il linguaggio di programmazione scelto per lo sviluppo di un software influisce sulla presenza di Community Smells, ovvero i modelli di caratteristiche organizzative e socio-tecniche non ottimali che possono portare a problemi tangibili nei team di sviluppo [15] all'interno del prodotto. Lo scopo è stato fornire una comprensione più approfondita dell'eventuale presenza di una correlazione tra il linguaggio di programmazione scelto per lo sviluppo e la presenza di Community Smells. La prospettiva è quella dei project manager, interessati a far utilizzare al team di sviluppo non solo il linguaggio più adatto al contesto funzionale, ma anche un linguaggio che contribuisca ad una maggiore qualità del codice finale.

3.1 Ipotesi e Domande di Ricerca

L'ipotesi di ricerca alla base di questo lavoro è la seguente:

La scelta del linguaggio di programmazione con cui sviluppare un prodotto può avere un impatto sulle attività di collaborazione e comunicazione, facendo emergere o attenuando i Community Smells.

Per facilitare il raggiungimento dell'obiettivo e per studiare meglio l'ipotesi precedentemente descritta, il processo di ricerca è stato strutturato in quattro fasi e, di conseguenza, nelle seguenti domande di ricerca:

- RQ₁** *Quali sono i linguaggi di programmazione maggiormente utilizzati nello sviluppo di software open-source su GITHUB?*
- RQ₂** *Quanto sono grandi (in termini di contributors) i team di sviluppo su GITHUB per i linguaggi individuati?*
- RQ₃** *Quali Community Smells sono presenti nelle repository dei datasets ottenuti?*
- RQ₄** *Il linguaggio di programmazione scelto per lo sviluppo di un prodotto influenza la presenza di Community Smells?*

La prima domanda di ricerca vuole essere un'analisi preliminare. Essa mira a fornire una panoramica di quali sono i linguaggi di programmazione maggiormente utilizzati nelle repository di GITHUB utilizzando GOOGLE BIGQUERY. Come la prima, la seconda domanda di ricerca mira a fornire una panoramica, ma a livello di team di sviluppo. L'obiettivo è studiare quanto siano effettivamente grandi i team di sviluppo (in termini di contributors, ossia sviluppatori che hanno realmente contribuito alla scrittura del codice) di GITHUB per i linguaggi individuati dalla domanda **RQ₁**. La terza domanda di ricerca mira invece a raccogliere e analizzare informazioni riguardo problemi di collaborazione e comunicazione all'interno delle repository attraverso l'utilizzo dei Community Smells, ovvero i modelli di caratteristiche organizzative e socio-tecniche non ottimali che possono portare a problemi tangibili nei team di sviluppo [15], e attraverso l'utilizzo di metriche socio-tecniche. L'ultima domanda di ricerca nasce dalla necessità di formalizzare i problemi sociali discussi in precedenza utilizzando uno studio statistico. In questa fase, attraverso la regressione lineare multipla, i linguaggi di programmazione sono correlati a un sottoinsieme di Community Smells che rappresentano meglio i problemi di collaborazione e comunicazione. Tale sottoinsieme di Community Smells è composto da: *Organizational Silo Effect*, *Black-cloud Effect*, *Prima-donnas Effect*, *Sharing Villainy*, *Organizational Skirmish*, *Solution Defiance*, *Radio Silence*, *Truck Factor Smell*, *Unhealthy Interaction* e *Toxic Communication*.

3.2 Contesto di Studio

Il contesto dello studio era costituito da linguaggi di programmazione, Community Smells e repository open-source. Nella Figura 3.1, è stato riassunto il processo di ricerca per lo studio quantitativo. Per le prime due domande di ricerca si necessitava di calcolare le informazioni tecniche delle repository GITHUB, quali linguaggi di programmazione e numero di contributors.

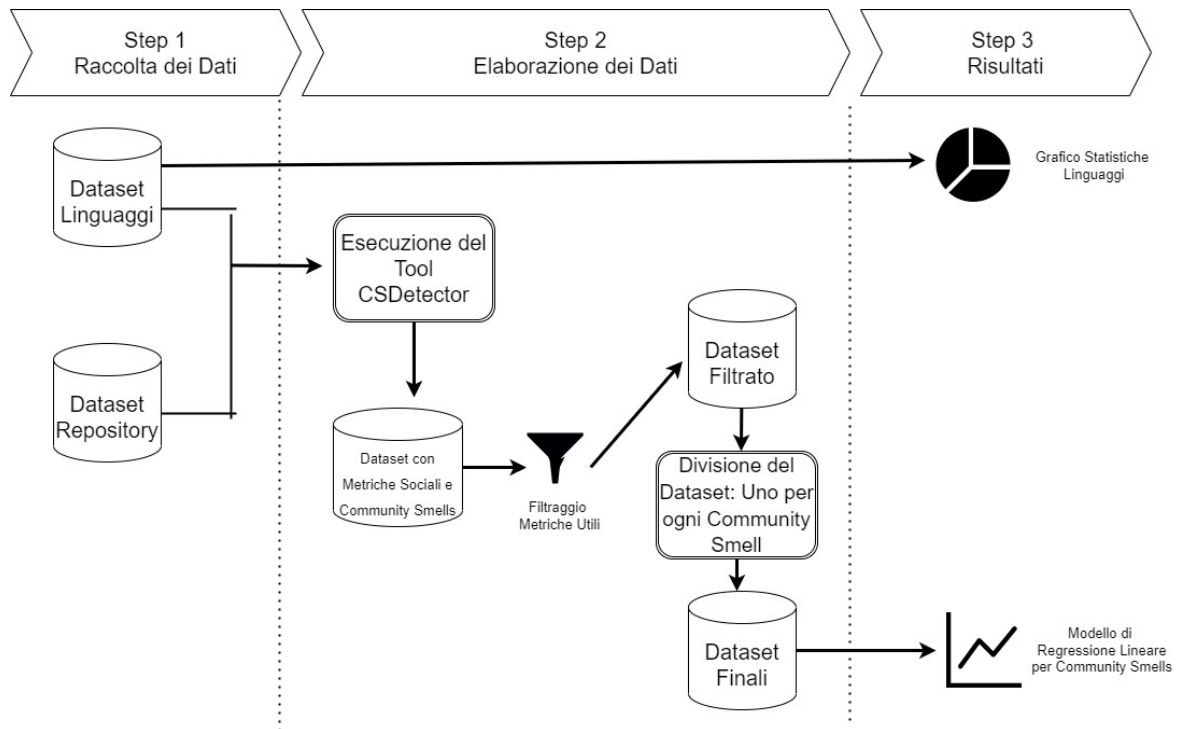


Figura 3.1: Panoramica del processo di ricerca

Il primo passo è stato reperire informazioni circa i linguaggi di programmazione maggiormente utilizzati nei progetti open-source. Nel dettaglio, sono stati selezionati i primi dieci linguaggi più utilizzati in repository open-source su GITHUB e che tra loro presentano caratteristiche diverse.

Il secondo passo è stato ottenere dei datasets di repository per ogni linguaggio di programmazione individuato al primo passo, e il numero di contributors per ogni repository.

Per ottenere le informazioni appena elencate è stato utilizzato GOOGLE BIGQUERY, un data warehouse cloud di Google che consente di archiviare e recuperare dati utilizzando un ambiente SQL.

Per la terza domanda di ricerca, si necessitava di reperire informazioni riguardo la presenza di Community Smells nelle repository presenti nei nostri datasets. Pertanto, si è deciso di utilizzare il tool CSDetector, descritto nella sezione 2.3 che, per ogni repository data in input restituisce i Community Smells presenti e delle metriche socio-tecniche.

I Community Smells e alcune metriche socio-tecniche restituite in output dal tool sono state riportate nel dataset delle repository.

Dopodiché è stato eseguito uno studio sulle diverse metriche socio-tecniche restituite in output dal tool CSDetector, così da individuare quali potevano essere effettivamente utili per lo studio inerente alla quarta domanda di ricerca.

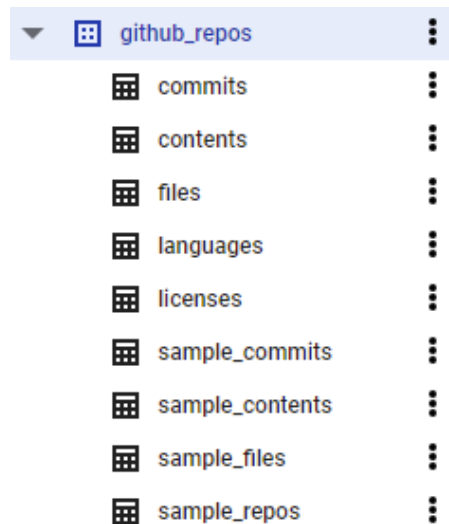


Figura 3.2: Struttura set di dati di GH ARCHIVE.

Il dataset costruito, quindi, è stato filtrato attraverso l’eliminazione delle metriche non rilevanti ai fini dello studio, ed è stato suddiviso in dieci datasets separati: uno per ogni Community Smell.

Infine, è stato costruito un modello di Regressione Lineare Multipla che, presi in input linguaggio di programmazione, metriche socio-tecniche e un Community Smell, ha permesso di studiare se effettivamente il linguaggio di programmazione ha influenza sulla presenza del Community Smell.

3.3 RQ₁ - Linguaggi di Programmazione su GITHUB

RQ₁. Quali sono i linguaggi di programmazione maggiormente utilizzati nello sviluppo di software open-source su GITHUB?

Per rispondere a RQ₁ è stata condotta un’indagine per individuare i linguaggi di programmazione maggiormente utilizzati nelle repository open-source presenti su GITHUB. Per ottenere queste informazioni lo strumento utilizzato è stato GOOGLE BIGQUERY¹: un servizio Web RESTful che permette un’analisi interattiva di grandi set di dati che lavora insieme a Google Storage, utilizzando un ambiente SQL. Tra i molteplici set di dati pubblici che è possibile interrogare su GOOGLE BIGQUERY è presente anche GH ARCHIVE², ossia l’intero archivio di repository pubbliche di GITHUB aggiornato automaticamente ogni ora. Dopo aver configurato l’ambiente, il primo passo è stato aggiungere al nostro progetto l’accesso

¹<https://console.cloud.google.com/>

²<https://www.gharchive.org/>

a *“bigquery-public-data.github_repos”*, ossia il set di dati di GH ARCHIVE la cui struttura è riportata nella Figura 3.2. Si può notare che le tabelle interrogabili sono: *commits*, *contents*, *files*, *languages*, *licenses*, *sample_commits*, *sample_contents*, *sample_files* e *sample_repos*. Dovendo recuperare l’informazione *“i linguaggi di programmazione maggiormente utilizzati”*, si è andati diretti nell’interrogare la Tabella *“bigquery-public-data.github_repos.languages”*.

La Figura 3.3 mostra la query eseguita su GOOGLE BIGQUERY per ottenere il numero di repository presenti su GITHUB per ogni linguaggio di programmazione: nella query più interna vengono estratte tutte le repository e tutti i linguaggi di programmazione utilizzati in esse, nella query più esterna le repository vengono raggruppate per linguaggio con *rank=1*. È importante notare che il codice sorgente in ogni repository può essere scritto in più linguaggi di programmazione, quindi la condizione *“rank=1”* ci ha permesso di considerare come linguaggio di programmazione quello maggiormente presente all’interno del codice.

Dal risultato di questa query è stato ottenuto un dataset composto da due colonne: *numberOfProject* e *Language*. Ciò ha permesso di estrarre i linguaggi di programmazione maggiormente utilizzati che sono poi stati classificati, al fine di utilizzare linguaggi di tipologia diversa, in base alle seguenti caratteristiche:

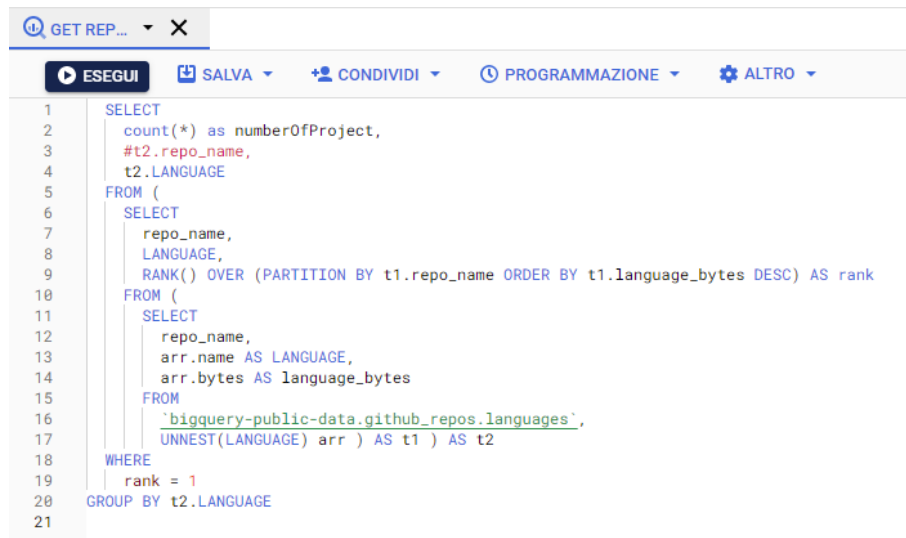
- **Paradigma di Programmazione:** procedurale, scripting, funzionale.
- **Compilazione:** statica, dinamica.
- **Tipizzazione:** forte, debole.
- **Memoria:** gestita, non gestita.

L’obiettivo è stato ottenere 10 linguaggi di programmazione diversi tra loro e maggiormente utilizzati su GITHUB.

3.4 RQ₂ - Contributors in Repository GITHUB

RQ₂. Quanto sono grandi (in termini di contributors) i team di sviluppo su GITHUB per i linguaggi individuati?

I contributors sono tutte quelle persone che contribuiscono alla stesura del codice presente in una repository su GITHUB. Per rispondere alla seconda domanda lo strumento utilizzato è stato GOOGLE BIGQUERY, proprio come per la prima domanda. Riprendendo la composizione del set di dati riportato nella Figura 3.2 è evidente che non c’è una Tabella che permette di



```

1  SELECT
2    count(*) as numberOfProject,
3    #t2.repo_name,
4    t2.LANGUAGE
5  FROM (
6    SELECT
7      repo_name,
8      LANGUAGE,
9      RANK() OVER (PARTITION BY t1.repo_name ORDER BY t1.language_bytes DESC) AS rank
10   FROM (
11     SELECT
12       repo_name,
13       arr.name AS LANGUAGE,
14       arr.bytes AS language_bytes
15     FROM
16       `bigquery-public-data.github_repos.languages`,
17       UNNEST(LANGUAGE) arr ) AS t1 ) AS t2
18   WHERE
19     rank = 1
20   GROUP BY t2.LANGUAGE
21

```

Figura 3.3: Query eseguita su GOOGLE BIGQUERY per ottenere le repository per linguaggio.

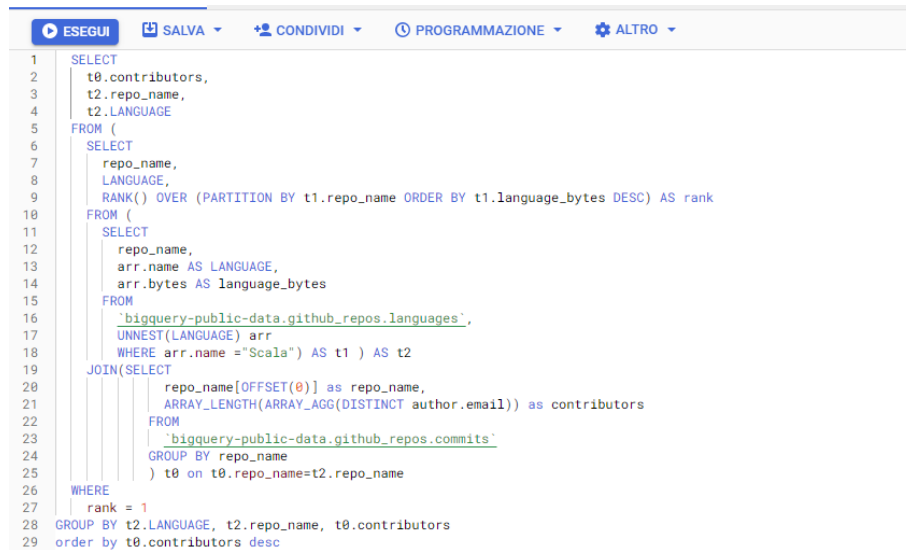
ottenere in maniera diretta i contributors di una repository. Pertanto, è stata costruita una query più complessa partendo da quella utilizzata in precedenza per recuperare i linguaggi di programmazione (mostrata nella Figura 3.3). La query costruita è mostrata nella Figura 3.4. Per ottenere il numero di contributor è stato deciso di interrogare la Tabella dei commits “*bigquery-public-data.github_repos.commits*”, dando per certo che ogni persona che abbia eseguito un commit nella repository abbia sicuramente contribuito alla stessa. La query è stata costruita andando a recuperare tutte le repository per un linguaggio di programmazione e, data la repository, sono stati recuperati tutti i diversi autori che hanno eseguito almeno un commit (query interne). Nella query più esterna, viene eseguito un filtraggio per rank=1, si raggruppa per linguaggio, nome della repository e contributors, e infine si ordina in ordine decrescente per numero di contributors. Nell’Immagine è mostrata la query costruita per il linguaggio “Scala”, ma la stessa è stata eseguita per tutti i dieci linguaggi ottenuti come risposta alla domanda RQ₁ posta nella sezione 3.3.

3.5 RQ₃ - Esecuzione del tool CSDetector

RQ₃. Quali Community Smells sono presenti nelle repository dei datasets ottenuti?

Per rispondere a RQ₃ è stato eseguito il tool CSDetector su 200 repository selezionate dai 10 datasets (20 repository per ogni linguaggio) ottenuti alla domanda RQ₂.

Il primo passo è stato configurare l’ambiente di esecuzione. Le configurazioni necessarie sono state:



```

1  SELECT
2    t0.contributors,
3    t2.repo_name,
4    t2.LANGUAGE
5  FROM (
6    SELECT
7      repo_name,
8      LANGUAGE,
9      RANK() OVER (PARTITION BY t1.repo_name ORDER BY t1.language_bytes DESC) AS rank
10   FROM (
11     SELECT
12       repo_name,
13       arr.name AS LANGUAGE,
14       arr.bytes AS language_bytes
15     FROM
16       `bigquery-public-data.github_repos.languages`,
17     UNNEST(LANGUAGE) arr
18     WHERE arr.name = "Scala") AS t1 ) AS t2
19   JOIN(SELECT
20     repo_name[OFFSET(0)] as repo_name,
21     ARRAY_LENGTH(ARRAY_AGG(DISTINCT author.email)) as contributors
22    FROM
23     `bigquery-public-data.github_repos.commits`
24    GROUP BY repo_name
25    ) t0 on t0.repo_name=t2.repo_name
26   WHERE
27     rank = 1
28   GROUP BY t2.LANGUAGE, t2.repo_name, t0.contributors
29   order by t0.contributors desc

```

Figura 3.4: Query eseguita su GOOGLE BIGQUERY per ottenere i contributors di ogni repository.

- Windows 10, già installato sulla macchina.
- Python 3.8.3
- Java 8.0.231
- PowerShell 7.0.1
- SentiStrength³
- ConvoKit⁴
- PyCharm, era consigliata nel ReadMe della repository⁵ l'installazione di VisualStudio Code 1.45.1, ma per comodità è stato deciso di utilizzare PyCharm.

Dopo aver installato tutto il necessario è stata aperta la PowerShell in modalità amministratore ed è stato eseguito il comando *installModules.ps1*, che ha creato un ambiente venv e installato tutti i moduli necessari con relative versioni corrette. In particolare sono stati installati i seguenti pacchetti:

- wheel==0.34.2
- networkx==2.4
- pandas

³<http://sentistrength.wlv.ac.uk/jkpop/>

⁴<https://convokit.cornell.edu/documentation/tutorial.html>

⁵<https://github.com/Nuri22/csDetector/blob/master/README.md>

- **matplotlib==3.2.1**
- **gitpython==3.1.2**
- **requests==2.23.0**
- **pyyaml==5.3.1**
- **progress==1.5**
- **python-dateutil==2.8.1**
- **sentistrength==0.0.8**
- **joblib==0.17.0**
- **scikit-learn==0.23.2**
- **imblearn==0.0**
- **xlrd==2.0.1**
- **openpyxl==3.0.9**
- **flask==2.1.2**
- **convokit==2.5.3**

A questo punto è stato attivato l'ambiente tramite il comando *.venv/Scripts* e costruito il comando da eseguire. Il comando da eseguire è costituito da diverse istruzioni:

- **python devNetwork.py**, che indica il file da avviare.
- **-p "accessToken"**, token identificativo del proprio account GITHUB.
- **-r "linkRepository"**, link della repository GITHUB su cui eseguire il tool (senza .git alla fine del link).
- **-s "path SentiStrength"**, path del proprio pc in cui risiede SentiStrength.
- **-o "path Output"**, path del proprio pc in cui si vogliono salvare i risultati restituiti dal tool CSDETECTOR.

È possibile inserire anche due comandi opzionali per determinare la data in cui si vuole iniziare l'analisi di un progetto e per accedere a GOOGLE CLOUD API. Non essendo rilevanti per lo studio in corso, questi due comandi non sono stati inseriti nel comando eseguibile.

Di seguito un esempio di comando eseguibile (al netto dell'accessToken di GITHUB, essendo un'informazione strettamente personale):

```
python devNetwork.py -p "accessToken" -r "https://github.com/mozilla-b2g/kernel_flatfish" -s
"C:/Users/Alice Vidoni/Desktop/csDetector-CR_2-web_service/senti/" -o "C:/Users/Alice Vidoni/Desktop
/Output_CSDetector"
```

Purtroppo, il tool CSDetector non gestisce tante tipologie di errore, ad esempio accesso ad array vuoti, divisioni per 0, ecc. e pertanto molte esecuzioni sono fallite. Ciò ha comportato che per tutti i linguaggi non sono stati ottenuti risultati per tutte le repository con precisamente 33 contributors, ma sono stati ottenuti risultati per repository con un range tra 30 e 50 contributors.

L'esecuzione del comando sopra esplicitato restituisce nel terminale della PowerShell i Community Smells presenti in quella repository e salva nel path indicato (-o) una cartella con dei file excel e grafici riguardanti i commit e dettagli su di essi, e un file excel con delle metriche socio-tecniche calcolate nel progetto. In particolare le metriche restituite sono riportate nella Tabella 3.1.

Dopo aver ottenuto risultati inerenti a 200 repository (20 repository per ognuno dei 10 linguaggi) è stato eseguito uno studio sulle metriche ottenute in output, così da determinare quali di queste sarebbero state utili per la fase successiva dello studio: la costruzione di modelli di regressione lineare multipla.

Tabella 3.1: Metriche Socio-Tecniche restituite da CSDetector

Categoria	Metrica	Descrizione
	NoD	Number of Developers
	NAD	Number of Active Days
	NCD	Number of Commits
	SDC	Standard Deviation of Commit
	NCD	Number of Code Developers
	PCD	Percentage of Core Developers
	NSD	Number of Sponsored Developers
Continua alla prossima pagina		

Tabella 3.1 – continua dalla pagina precedente

Categoria	Metrica	Descrizione
Developer Contributions metrics	PSD	Percentage of Sponsored Developers
	NPR	Total Number of Pull Requests
	SAPR	Standard Deviation of author for Pull Requests
	ANAPR	Average Number of authors for Pull Requests
	NI	Number of Issues
	SDAI	Standard Deviation of authors for issue report
	ANAI	Average Number of authors for issue report
Social Network Analysis metrics	GCD	Graph Degree Centrality
	SDD	Standard Deviation of a graph Degree centrality in a project
	GBC	Graph Betweenness Centrality
	GCC	Graph Closeness Centrality
	ND	Network Density
Community metrics	NC	Number of Communities
	ACC	Average of Commits for Community
	SCC	Standard Deviation of Commits for Community
	ADC	Average Number of Developers for Community
	SDC	Standard Deviation of Developers for Community
Geographic Dispersion metrics	TZ	Number of Time Zones
	ACZ	Average of Commits for Time Zone
	SCZ	Standard Deviation of Commits for Time Zones
	ADZ	Average Number of Developers for Time Zone
	SDZ	Standard Deviation of Developers for Time Zones
Formality metrics	NR	Number of Releases in a project
	PCR	Percentage of Commits for Release
	FN	Formal Network
	ADPR	Average number of days for Pull Request
	ADI	Average Number of Days for issue report
Truck Number and Community Members metrics	BFN	Bus Factor Number
	TFN	Global TruckNumber

Continua alla prossima pagina

Tabella 3.1 – continua dalla pagina precedente

Categoria	Metrica	Descrizione
Communication metrics	TFC	Truck Factor Coverage
	ANCPR	Average number of comments for Pull Request
	SCPR	Standard Deviation of commits for Pull Request
	NCI	Number Comments in issues
	ANCI	Average number of comments for issue report
	SDCI	Standard Deviation of Comments
		Count for issue report
	RTCPR	Ratio of toxic comments in Pull discussions
	RTCI	Ratio of toxic comments in issue discussions
	RPCPR	Ratio of polite comments in Pull Requests discussions
Sentiment Analysis metrics	RPCI	Ratio of polite comments in issue discussions
	RINC	Ratio of issues with negative sentiments
	RNSPRC	Ratio of negative sentiments in Pull Requests comments
	RAWPR	Ratio of anger words in Pull Requests comments
	RAWI	Ratio of anger words in issue discussions
	ACCL	Average Communication Comments Length

3.6 RQ₄ - Modelli di Regressione Lineare

RQ₄. Il linguaggio di programmazione scelto per lo sviluppo di un prodotto influenza la presenza di Community Smells?

Per lavorare al quesito RQ₄ è stato definito un modello statistico che mette in relazione il linguaggio di programmazione e la frequenza dei Community Smells nelle repository del dataset.

È stato costruito il dataset da utilizzare unendo il dataset principale (con le informazioni generali delle repository open-source, quali nomi delle repository, linguaggio di programmazione e contributors), con le informazioni ottenute in output dal tool (quindi presenza

o no di Community Smells nella repository e alcune metriche sociali quali CommitCount, DaysActive, BusFactorNumber). Il tutto per un totale di 200 repository.

3.6.1 Variabili Indipendenti

In base all'ipotesi dello studio, è stato considerato il fattore fondamentale dello studio come variabile indipendente:

- **Linguaggio di Programmazione**, indica il linguaggio di programmazione maggiormente utilizzato nel codice presente all'interno di una repository GITHUB.

3.6.2 Variabile di Risposta

L'obiettivo è stato quello di comprendere l'impatto di un linguaggio di programmazione sulla presenza di Community Smells. Per questo motivo, la variabile di risposta era rappresentata dal numero di Community Smells che il tool CSDETECTOR è in grado di restituire per ogni repository, ovvero *Organizational Silo Effect*, *Black-Cloud Effect*, *Prima-Donnas Effect*, *Sharing Villainy*, *Organizational Skirmish*, *Solution Defiance*, *Radio Silence*, *Truck Factor Smell*, *Unhealthy Interaction* e *Toxic Communication*.

Variabile di Risposta: istanze di Community Smells.

La variabile dipendente è il numero di istanze di dieci tipi di Community Smells, ossia *Organizational Silo Effect*, *Black-Cloud Effect*, *Prima-Donnas Effect*, *Sharing Villainy*, *Organizational Skirmish*, *Solution Defiance*, *Radio Silence*, *Truck Factor Smell*, *Unhealthy Interaction* e *Toxic Communication*, all'interno di una repository open-source

3.6.3 Variabili di Controllo

Mentre l'ipotesi riguardava la correlazione tra il linguaggio di programmazione utilizzato maggiormente nella repository e la presenza di Community Smells, è fondamentale notare che molti altri fattori legati alla repository potrebbero influenzare la variabile dipendente, e sono proprio le metriche socio-tecniche restituite dal tool CSDETECTOR che sono state inserite nella costruzione del dataset finale.

Questi ulteriori valori, quindi, sono stati considerati come variabili di controllo:

- **Contributors**, indica il numero totale di persone differenti che hanno eseguito almeno un commit nella finestra temporale che va dall'apertura della repository fino al giorno

di esecuzione del tool CSDETECTOR. Catolino et al. [12] hanno dimostrato che il numero di commit in una comunità può influenzare il numero di Community Smells.

- **CommitCount**, indica il numero totale di commit che sono stati eseguiti su una repository fino al giorno in cui il tool CSDETECTOR è stato eseguito. Nella maggior parte dei casi, un numero elevato di commit può indicare un'elevata attività nella community. Poiché non è raro che più sviluppatori lavorino sullo stesso modulo di codice o sullo stesso requisito funzionale, tale attività potrebbe corrispondere a un'elevata comunicazione e collaborazione, con un possibile impatto sul numero di Community Smells.
- **DaysActive**, indica da quanti giorni la repository risulta essere attiva, fino al giorno in cui il tool CSDETECTOR è stato eseguito.
- **BusFactorNumber**, è una metrica socio-tecnica, un fattore di rischio che indica il numero minimo di membri del team che devono abbandonare improvvisamente un progetto prima che il progetto si blocchi a causa della mancanza di personale esperto o competente, e quindi fallisca [51, 38, 52].

3.6.4 Costruzione del Modello Statistico

Le variabili dipendenti consistono nel numero di istanze di dieci tipi di Community Smells: *Organizational Silo Effect*, *Black-Cloud Effect*, *Prima-Donnas Effect*, *Sharing Villainy*, *Organizational Skirmish*, *Solution Defiance*, *Radio Silence*, *Truck Factor Smell*, *Unhealthy Interaction* e *Toxic Communication*. Per questo motivo sono stati costruiti dieci modelli di Regressione Lineare Multipla⁶, uno per ogni Community Smell considerato, tutti con le stesse variabili indipendenti e variabili di controllo.

Per ottenere i risultati dello studio e costruire il modello di Regressione Lineare Multipla è stata utilizzata la libreria Python `statsmodel` [53]. Questa libreria fornisce classi e funzioni per la stima di molti modelli statistici diversi, nonché per condurre test statistici ed esplorazione di dati statistici.

Un modello di regressione lineare accetta solo valori numerici, nel dataset, però, il linguaggio di programmazione era un valore testuale, quindi si è presentata necessità di categorizzare

⁶*Regressione Lineare Multipla* è un'estensione dell'analisi della correlazione e della regressione lineare semplice. È una tecnica statistica utilizzata per prevedere il risultato di una variabile in base al valore di due o più variabili. La variabile da prevedere è la variabile dipendente.

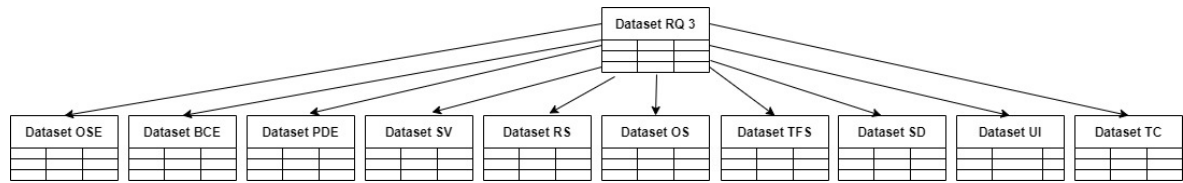


Figura 3.5: Suddivisione dataset per applicare Regressione Lineare.

questa variabile. Pertanto, ad ogni linguaggio di programmazione presente nel dataset è stato assegnato un numero progressivo simbolico.

Il secondo passo è stato suddividere il dataset in dieci dataset distinti, uno per ogni Community Smell, come mostrato in Figura 3.5. A seguito di questa operazione, quindi, ogni dataset presentava le seguenti colonne: *Community Smell*, *Linguaggio di Programmazione*, *Contributors*, *CommitCount*, *DaysActive*, *BusFactorNumber*.

Il passo successivo è stato costruire il modello di regressione lineare multipla in Python. Le funzioni utilizzate sono state:

- **sm.add_constant(x)**, per aggiungere la colonna di uno agli input, così che `statsmodel` calcoli l'intercept `b0`.
- **model = sm.OLS(y, x)**, rappresenta il modello di regressione basato sui minimi quadrati ordinari.
- **results = model.fit()**, chiamando `.fit()` si ottiene la variabile `results`, che è un'istanza della classe `statsmodels.regression.linear_model.RegressionResultsWrapper`. Questo oggetto contiene le informazioni necessarie sul modello di regressione.
- **results.summary()**, per ottenere la Tabella con i risultati della regressione lineare.

Questo capitolo illustra i risultati ottenuti per ogni domanda di ricerca e le conclusioni relative all'obiettivo principale.

4.1 Dataset: Statistiche di Sintesi e Informazioni Generali - RQ1

In questo paragrafo si fornirà la risposta alla prima domanda di ricerca.

RQ₁. Quali sono i linguaggi di programmazione maggiormente utilizzati nello sviluppo di software open-source su GITHB?

Come detto nella sezione 3.3, l'obiettivo di **RQ₁** è stato quello di ottenere una visione dei linguaggi di programmazione maggiormente utilizzati nelle repository open-source presenti su GITHUB. La Figura 4.1 mostra un grafico a torta che rappresenta i linguaggi di programmazione maggiormente utilizzati. È importante sottolineare che per rendere il grafico ben leggibile sono stati inseriti solo i primi 17 linguaggi in ordine di grandezza decrescente per numero di repository. A primo impatto si può notare che i più utilizzati sono JavaScript, Python, Java e PHP.

Nella Figura 4.2, vengono riportate le stesse informazioni discusse in precedenza, ma utilizzando dei numeri. Sull'asse delle ascisse abbiamo i linguaggi di programmazione, mentre sull'asse delle ordinate il numero di repository open-source presenti su GITHUB che hanno come linguaggio principale quello riportato sull'asse delle ascisse. Anche con questa

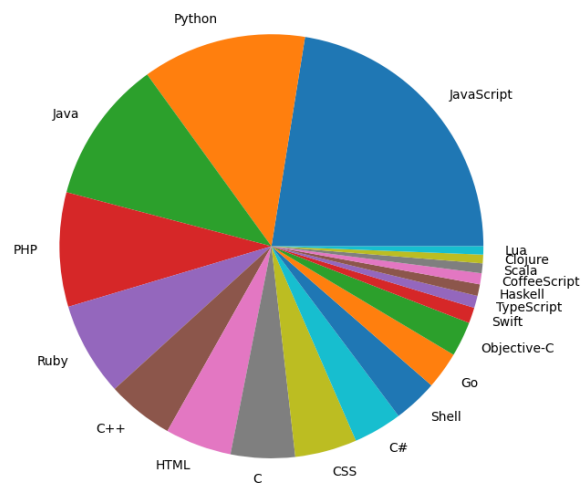


Figura 4.1: Linguaggi di Programmazione maggiormente utilizzati

tipologia di grafico è evidente che i linguaggi di programmazione maggiormente utilizzati sono gli stessi citati sopra.

A seguito dell'individuazione dei linguaggi maggiormente utilizzati e dell'analisi delle loro caratteristiche, viene riportata la Tabella 4.1, che contiene i dieci linguaggi che è stato deciso di utilizzare per procedere con lo studio.

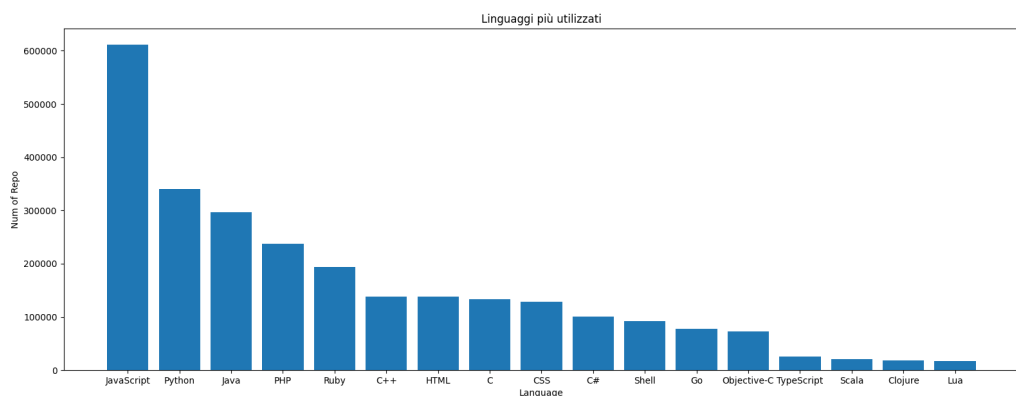


Figura 4.2: Linguaggi di Programmazione maggiormente utilizzati

Tabella 4.1: Linguaggi di programmazione e le loro caratteristiche.

Linguaggio	Paradigma di Programmazione	Compilazione	Tipizzazione	Memoria
C	Procedurale	Statica	Debole	Non Gestita
C#	Procedurale	Statica	Forte	Gestita
C++	Procedurale	Statica	Forte	Non Gestita
Java	Procedurale	Statica	Forte	Gestita
JavaScript	Scripting	Dinamica	Debole	Gestita
Python	Scripting	Dinamica	Forte	Gestita
PHP	Scripting	Dinamica	Debole	Gestita
Scala	Funzionale	Statica	Forte	Gestita
TypeScript	Scripting	Statica	Forte	Gestita
Go	Procedurale	Statica	Forte	Non Gestita

RQ₁: Sommario dei Risultati.

I linguaggi di programmazione maggiormente utilizzati su GITHUB e che presentano caratteristiche diverse tra loro sono: *Python, Java, PHP, C, C#, C++, Go, JavaScript, Scala e TypeScript*.

4.2 Dataset: Statistiche di Sintesi e Informazioni Generali - RQ2

In questo paragrafo si fornirà la risposta alla seconda domanda di ricerca.

RQ₂. Quanto sono grandi (in termini di contributors) i team di sviluppo su GITHB per i linguaggi individuati?

Come detto nella sezione 3.4, l'obiettivo di **RQ₂** è stato quello di ottenere una visione del numero di contributors delle repository open-source presenti su GITHUB per ogni linguaggio di programmazione individuato nella sezione 4.1. Dal risultato della query sono stati ottenuti dieci datasets composti ognuno da 3 colonne: numero di contributors, nome della repository e linguaggio di programmazione principale.

Al fine determinare le repository da utilizzare per l'esecuzione del tool CSDETECTOR è stata calcolata la media del numero di contributors per ogni linguaggio, come mostrato nel grafico a barre orizzontali nella Figura 4.3. Successivamente è stata calcolata la media delle medie del numero di contributors: 32,59.

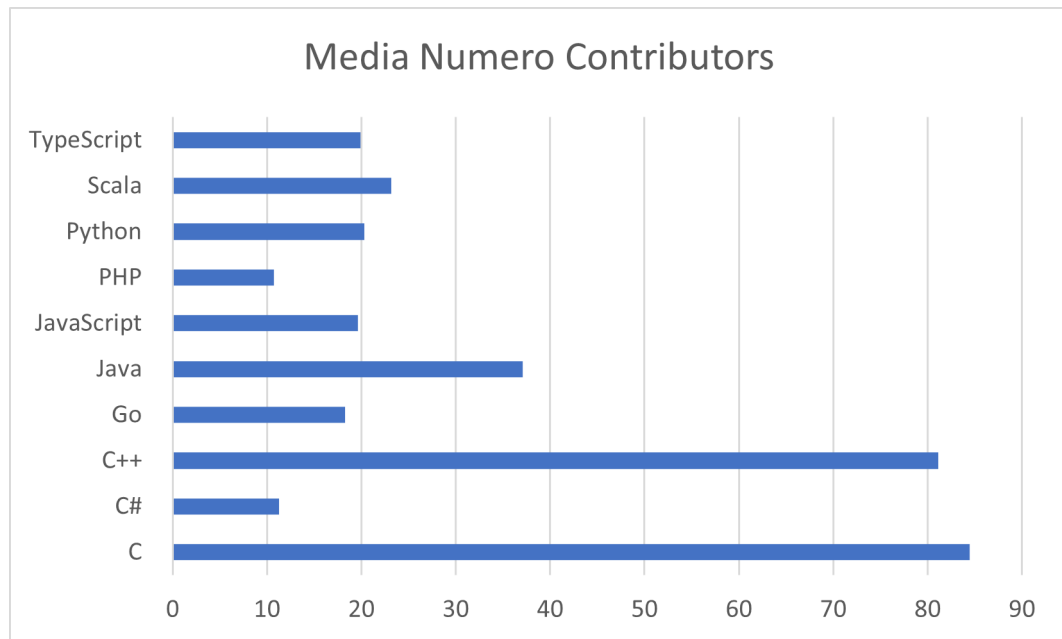


Figura 4.3: Media numero di contributors per ogni linguaggio.

Pertanto, abbiamo definito che al tool CSDetector dovevano essere passate in input repository con un numero di contributors maggiore o uguale a 33.

RQ₂: Sommario dei Risultati.

Il numero medio di contributors tra le repository dei vari linguaggi risulta essere molto diverso, pertanto è stata calcolata la media delle medie per ottenere il numero medio dei contributors, che ammonta a 33.

4.3 Dataset: Statistiche di Sintesi e Informazioni Generali - RQ3

In questo paragrafo si fornirà la risposta alla terza domanda di ricerca.

RQ₃. Quali Community Smells sono presenti nelle repository dei datasets ottenuti?

Come detto nella sezione 3.5, l'obiettivo di RQ₃ è stato quello di determinare quali Community Smells fossero presenti nelle repository del nostro dataset attraverso l'esecuzione del tool CSDetector per ogni repository.

Ogni esecuzione del tool ha restituito in output nel terminale i Community Smells presenti nella repository e un file excel con delle metriche socio-tecniche (riportate nella Tabella 3.1).

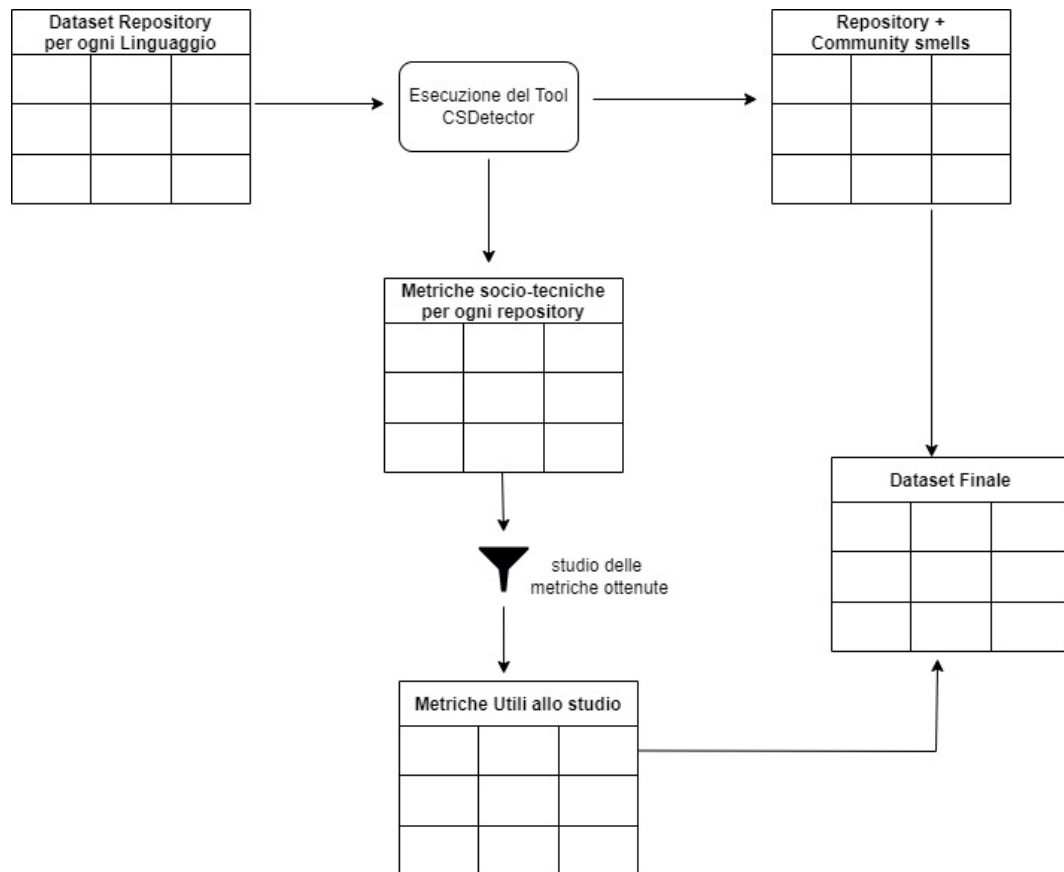


Figura 4.4: Panoramica della gestione dei risultati del Tool CSDetector

Dopo aver eseguito il tool CSDetector su tutte le 200 repository del datasets, è stato eseguito uno studio su quali tra le metriche restituite in output fossero utili per procedere con lo studio. Le metriche che abbiamo deciso di tenere in considerazione sono state:

- **CommitCount**, indica il numero totale di commit che sono stati eseguiti su una repository fino al giorno in cui il tool CSDetector è stato eseguito. Nella maggior parte dei casi, un numero elevato di commit può indicare un'elevata attività nella community. Poiché non è raro che più sviluppatori lavorino sullo stesso modulo di codice o sullo stesso requisito funzionale, tale attività potrebbe corrispondere a un'elevata comunicazione e collaborazione, con un possibile impatto sul numero di Community Smells.
- **DaysActive**, indica da quanti giorni la repository risulta essere attiva, fino al giorno in cui il tool CSDetector è stato eseguito.
- **BusFactorNumber**, è una metrica socio-tecnica, un fattore di rischio che indica il numero minimo di membri del team che devono abbandonare improvvisamente un

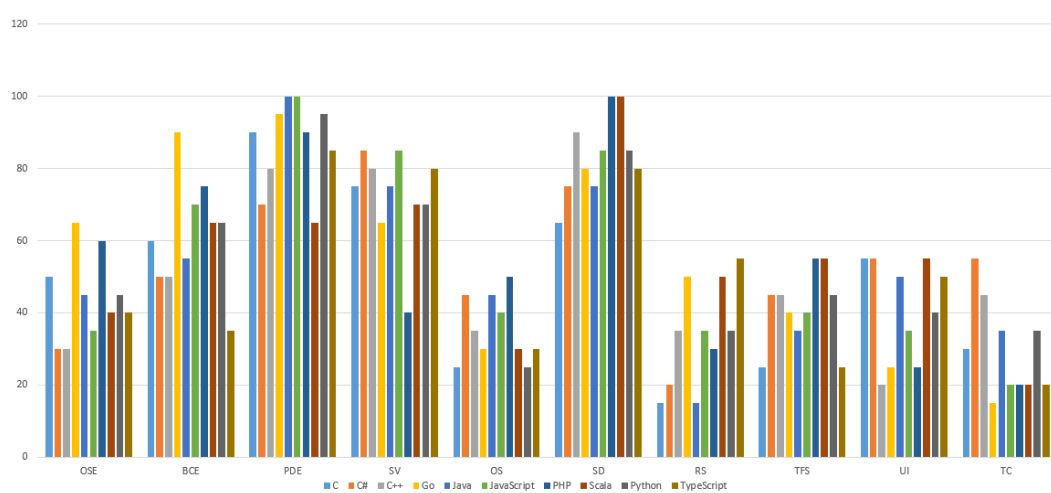


Figura 4.5: Percentuale di presenza di uno Smell nei Linguaggi di Programmazione

progetto prima che il progetto si blocchi a causa della mancanza di personale esperto o competente, e quindi fallisca [51, 38, 52].

L'individuazione di queste metriche ha permesso di completare la costruzione del dataset come mostrato nella Figura 4.4.

Le colonne inerenti alle metriche socio-tecniche sono state riportate esattamente come sono state restituite in output dal tool CSDETECTOR, mentre per quanto riguarda i Community Smells è stata inserita una colonna per ogni possibile Community Smell ed è stata poi popolata con 1 se lo Smell è stato restituito dal tool nel terminale di PowerShell (e quindi è presente nella repository), 0 altrimenti.

Il dataset finale quindi è composto dalle seguenti colonne: *RepositoryName*, *Language*, *Contributors*, *OSE*, *BCE*, *PDE*, *RS*, *SD*, *SV*, *OS*, *TFS*, *UI*, *TC*, *CommitCount*, *DaysActive*, *BusFactorNumber*.

La costruzione del dataset finale ha permesso di eseguire un primo studio statistico inerente alla presenza di Community Smells per ogni specifico linguaggio di programmazione. In particolare, la Figura 4.5 riporta un grafico a barre in cui sull'asse delle ascisse sono riportati i Community Smells e sull'asse delle ordinate un valore numerico che indica la percentuale di presenza dello Smell per ogni specifico linguaggio di programmazione. Per rendere i valori più leggibili li riportiamo anche sottoforma di Tabella nella figura 4.2. Come si può notare da entrambe le fonti, i Community Smells maggiormente presenti nei 10 linguaggi di programmazione sono *Prima-Donnas Effect (PDE)* e *Solution Defiance (SD)*.

Tabella 4.2: Presenza di Community Smells per Linguaggio

Linguaggio	OSE	BCE	PDE	SV	OS	SD	RS	TFS	UI	TC
C	50%	60%	90%	75%	25%	65%	15%	25%	55%	30%
C#	30%	50%	70%	85%	45%	75%	20%	45%	55%	55%
C++	30%	50%	80%	80%	35%	90%	35%	45%	20%	45%
Go	65%	90%	95%	65%	30%	80%	50%	40%	25%	15%
Java	45%	55%	100%	75%	45%	75%	15%	35%	50%	35%
JavaScript	35%	70%	100%	85%	40%	85%	35%	40%	35%	20%
PHP	60%	75%	90%	40%	50%	100%	30%	55%	25%	20%
Scala	40%	65%	65%	70%	30%	100%	50%	55%	55%	20%
Pyhton	45%	65%	95%	70%	25%	85%	35%	45%	40%	35%
TypeScript	40%	35%	85%	80%	30%	80%	55%	25%	50%	20%

RQ₃: Sommario dei Risultati.

È stato ottenuto il dataset finale con tutte le informazioni utili allo studio successivo, ed è stato determinato che i Community Smell maggiormente presenti nelle repository del dataset sono *PDE* ed *SD*.

4.4 Modelli di Regressione - RQ4

In questo paragrafo si fornirà la risposta alla quarta domanda di ricerca.

RQ₄. Il linguaggio di programmazione scelto per lo sviluppo di un prodotto influenza la presenza di Community Smells?

In questa sezione riportiamo i risultati che riguardano **RQ₄**. Lo scopo era costruire dieci modelli di regressione lineare multipla, uno per tipo di Community Smells, per studiare la correlazione tra il linguaggio di programmazione - più altri fattori di controllo - e il numero di Community Smells in una comunità di sviluppo. Nella Tabella 4.3 è riportato un riassunto dei dieci Community Smells coinvolti in questo studio. Per costruire i modelli di cui sopra è stata utilizzata la libreria `statsmodel` di Python. A seguire, una sezione dedicata all'analisi dei risultati ottenuti per ogni modello costruito. In ogni Immagine che mostra i risultati sono rappresentati sei valori *const*, *x1*, *x2*, *x3*, *x4*, *x5*. Questi valori rappresentano nel dettaglio:

- *const* -> Interceptor

Tabella 4.3: Panoramica sui community smells analizzati [18]

Community Smell	Definizione
Organizational Silo Effect	Si riferisce alla presenza di sottogruppi isolati e alla mancanza di comunicazione e collaborazione tra gli sviluppatori della comunità.
Black Cloud Effect	Eccessivo sovraccarico di informazioni dovuto alla mancanza di una comunicazione strutturata o di una governance della cooperazione.
Radio Silence	Si verifica un'elevata formalità di procedure regolari a causa dell'inefficiente organizzazione strutturale di una comunità.
Prima Donna	Ripetuto comportamento condiscendente, superiorità, disaccordo costante, mancanza di collaborazione da parte di uno o pochi membri.
Sharing Villainy	Mancanza di attività di scambio di informazioni di qualità.
Organizational Skirmish	Causato da un disallineamento tra diversi livelli di competenza e canali di comunicazione tra unità di sviluppo o individui coinvolti nel progetto.
Solution Defiance	La comunità di sviluppo presenta diversi livelli di background culturale ed esperienziale e queste variazioni portano alla divisione della comunità in sottogruppi simili con opinioni completamente contrastanti sulle decisioni tecniche o socio-tecniche da prendere.
Truck Factor Smell	La maggior parte delle informazioni e delle conoscenze sul progetto sono concentrate in uno o pochi sviluppatori.
Unhealthy Interaction	Le discussioni tra gli sviluppatori sono lente, leggere, brevi e/o contengono conversazioni scadenti.
Toxic Communication	Le comunicazioni tra gli sviluppatori sono soggette a conversazioni tossiche e sentimenti negativi contenenti opinioni spiacevoli, di rabbia o addirittura contrastanti su vari argomenti di cui le persone discutono.

- x1 -> Linguaggio di Programmazione
- x2 -> Contributors
- x3 -> CommitCount
- x4 -> DaysActive
- x5 -> BusFactorNumber

I valori restituiti dalla libreria di Python e principalmente rilevanti per lo studio sono:

- **Covariance Type**, quasi sempre “nonrobust”, il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. La covarianza mostra come due variabili si muovono l'una rispetto all'altra. Se >0 , le variabili si muovono nella stessa direzione, se <0 allora funzionano in direzione opposta. Importante notare che la covarianza è diversa dalla correlazione, infatti essa non fornisce la forza della relazione ma solo la direzione del movimento.

- **R-square**, è il coefficiente di determinazione che indica la percentuale della variabilità dei dati spiegata dalle variabili indipendenti selezionate (sempre compreso tra 0 e 1). Questo valore più si avvicina all'uno, maggiore è la dipendenza tra le variabili.
- **Prob(F-Statistics)**, il test F fornisce un modo per verificare tutte le variabili indipendenti insieme se qualcuna di esse è correlata alla variabile dipendente. Se la Prob(F-statistic) è maggiore di 0.05, non c'è evidenza di una relazione tra una qualsiasi variabile indipendente e la variabile dipendente. Se è inferiore a 0.05, possiamo affermare che c'è almeno una variabile che è significativamente correlata all'output. Tuttavia, in alcuni casi il prob(F-statistic) può essere maggiore di 0.05 ma una delle variabili indipendenti mostra una forte correlazione. Ciò è dovuto al fatto che ogni t-test viene effettuato con una serie diversa di dati, mentre il test F verifica l'effetto combinato di tutte le variabili.
- **P>|t|**, la colonna t fornisce i valori t corrispondenti a ciascuna variabile indipendente. Le statistiche T sono utilizzate per calcolare i valori p. In genere, quando il valore p è inferiore a 0.05, indica una forte evidenza contro l'ipotesi nulla, che afferma che la variabile indipendente corrispondente non ha alcun effetto sulla variabile dipendente.

4.4.1 Modello di Regressione per Community Smell: OSE

L'Immagine 4.6 riporta i dettagli relativi al modello statistico costruito per lo smell *Organizational Silo Effect*. Il modello ha restituito come COVARIANCETYPE il valore "nonrobust", il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.056. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, tuttavia essendo leggermente maggiore indica che qualche variabile indipendente contribuisce alla variabilità della variabile dipendente (come conferma anche il valore di PROB(F-STATISTIC) pari a 0.0458). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna P>|T|. L'unico valore di questa colonna <0.05 è quello relativo al *BusFactorNumber* (x5), che corrisponde a 0.002. Da ciò si evince che *BusFactorNumber* è l'unica variabile indipendente che influenza la presenza del *Community Smell OSE*.

4.4.2 Modello di Regressione per Community Smell: BCE

L'Immagine 4.7 riporta i dettagli relativi al modello statistico costruito per lo smell *Black-cloud Effect*. Il modello ha restituito come COVARIANCETYPE il valore "nonrobust", il che

OLS Regression Results						
=====						
Dep. Variable:	y	R-squared:	0.056			
Model:	OLS	Adj. R-squared:	0.032			
Method:	Least Squares	F-statistic:	2.308			
Date:	Tue, 26 Jul 2022	Prob (F-statistic):	0.0458			
Time:	16:22:37	Log-Likelihood:	-137.93			
No. Observations:	200	AIC:	287.9			
Df Residuals:	194	BIC:	307.7			
Df Model:	5					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

const	-0.2782	0.270	-1.031	0.304	-0.811	0.254
x1	-0.0018	0.012	-0.146	0.884	-0.026	0.022
x2	-0.0014	0.004	-0.338	0.736	-0.010	0.007
x3	5.326e-06	2.22e-05	0.240	0.811	-3.85e-05	4.91e-05
x4	1.508e-05	3.21e-05	0.470	0.639	-4.81e-05	7.83e-05
x5	0.7834	0.252	3.104	0.002	0.286	1.281
=====						
Omnibus:	1170.979	Durbin-Watson:	1.898			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	30.924			
Skew:	0.148	Prob(JB):	1.93e-07			
Kurtosis:	1.096	Cond. No.	3.18e+04			
=====						

Figura 4.6: Risultati ottenuti dal modello per il Community Smell OSE

significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.262. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, tuttavia essendo leggermente maggiore indica che qualche variabile indipendente contribuisce alla variabilità della variabile dipendente (come conferma anche il valore di PROB(F-STATISTIC) pari a 0.000000000154). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna P>|T|. I valori di questa colonna <0.05 sono quelli relativi a *CommitCount* (x3) che corrisponde a 0.000, *DaysActive* (x4) che corrisponde a 0.001 e *BusFactorNumber* (x5) che corrisponde a 0.000. Da ciò si evince che *CommitCount*, *DaysActive* e *BusFactorNumber* sono le variabili indipendenti che influenzano la presenza del Community Smell BCE.

4.4.3 Modello di Regressione per Community Smell: PDE

L'Immagine 4.8 riporta i dettagli relativi al modello statistico costruito per lo smell *Prima-Donnas Effect*. Il modello ha restituito come COVARIANCETYPE il valore "nonrobust", il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.023. Quando questo valore è 0 significa che le variabili

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.262			
Model:	OLS	Adj. R-squared:	0.243			
Method:	Least Squares	F-statistic:	13.81			
Date:	Tue, 26 Jul 2022	Prob (F-statistic):	1.54e-11			
Time:	16:22:37	Log-Likelihood:	-109.28			
No. Observations:	200	AIC:	230.6			
Df Residuals:	194	BIC:	250.3			
Df Model:	5					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	0.0101	0.234	0.043	0.966	-0.451	0.471
x1	-0.0119	0.011	-1.112	0.268	-0.033	0.009
x2	-0.0070	0.004	-1.926	0.056	-0.014	0.000
x3	-8.995e-05	1.92e-05	-4.675	0.000	-0.000	-5.2e-05
x4	9.11e-05	2.78e-05	3.280	0.001	3.63e-05	0.000
x5	0.8738	0.219	3.995	0.000	0.442	1.305
Omnibus:	58.873	Durbin-Watson:	1.963			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	20.595			
Skew:	-0.573	Prob(JB):	3.37e-05			
Kurtosis:	1.924	Cond. No.	3.18e+04			

Figura 4.7: Risultati ottenuti dal modello per il Community Smell BCE

indipendenti date in input al modello non spiegano la variabilità, essendo molto vicino allo 0 c'è una buona possibilità che nessuna variabile indipendente contribuisca alla variabilità della variabile dipendente (come conferma anche il valore di $\text{PROB}(\text{F-STATISTIC})$ pari a 0.468, molto vicino a 0.5). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna $P > |t|$. In questa colonna non risultano esserci valori < 0.05 . Da ciò si evince che *nessuna variabile indipendente influenza la presenza del Community Smell PDE*.

4.4.4 Modello di Regressione per Community Smell: SV

L'Immagine 4.9 riporta i dettagli relativi al modello statistico costruito per lo smell *Sharing Villainy*. Il modello ha restituito come `COVARIANCETYPE` il valore "nonrobust", il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di `R-SQUARED` è 0.023. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, essendo molto vicino allo 0 c'è una buona possibilità che nessuna variabile indipendente contribuisca alla variabilità della variabile dipendente (come conferma anche il valore di $\text{PROB}(\text{F-STATISTIC})$ pari a 0.461, molto vicino a 0.5). Per individuare quale variabile indipendente influenza la presenza della variabile

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.023			
Model:	OLS	Adj. R-squared:	-0.002			
Method:	Least Squares	F-statistic:	0.9217			
Date:	Tue, 26 Jul 2022	Prob (F-statistic):	0.468			
Time:	16:22:37	Log-Likelihood:	-56.630			
No. Observations:	200	AIC:	125.3			
Df Residuals:	194	BIC:	145.1			
Df Model:	5					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	1.1522	0.180	6.410	0.000	0.798	1.507
x1	0.0033	0.008	0.398	0.691	-0.013	0.019
x2	-0.0022	0.003	-0.784	0.434	-0.008	0.003
x3	1.324e-05	1.48e-05	0.896	0.372	-1.59e-05	4.24e-05
x4	-2.329e-05	2.13e-05	-1.091	0.277	-6.54e-05	1.88e-05
x5	-0.1725	0.168	-1.026	0.306	-0.504	0.159
Omnibus:	96.254	Durbin-Watson:	1.877			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	257.280			
Skew:	-2.251	Prob(JB):	1.36e-56			
Kurtosis:	6.256	Cond. No.	3.18e+04			

Figura 4.8: Risultati ottenuti dal modello per il Community Smell PDE

dipendente sono stati analizzati i valori ottenuti nella colonna $P > |t|$. In questa colonna non risultano esserci valori < 0.05 . Da ciò si evince che *nessuna variabile indipendente influenza la presenza del Community Smell SV*.

4.4.5 Modello di Regressione per Community Smell: OS

L'Immagine 4.10 riporta i dettagli relativi al modello statistico costruito per lo smell *Organizational Skirmish*. Il modello ha restituito come COVARIANCETYPE il valore "nonrobust", il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.048. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, tuttavia essendo leggermente maggiore indica che qualche variabile indipendente contribuisce alla variabilità della variabile dipendente (come conferma anche il valore di PROB(F-STATISTIC) pari a 0.0854). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna $P > |t|$. L'unico di questa colonna < 0.05 è quello relativo a *Contributors* (x2) che corrisponde a 0.033. Da ciò si evince che *Contributors* è l'unica variabile indipendente che influenza la presenza del Community Smell OS.

OLS Regression Results						
=====						
Dep. Variable:	y	R-squared:	0.023			
Model:	OLS	Adj. R-squared:	-0.002			
Method:	Least Squares	F-statistic:	0.9326			
Date:	Tue, 26 Jul 2022	Prob (F-statistic):	0.461			
Time:	16:22:37	Log-Likelihood:	-120.16			
No. Observations:	200	AIC:	252.3			
Df Residuals:	194	BIC:	272.1			
Df Model:	5					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

const	0.9395	0.247	3.804	0.000	0.452	1.427
x1	-0.0086	0.011	-0.762	0.447	-0.031	0.014
x2	0.0024	0.004	0.627	0.531	-0.005	0.010
x3	1.9e-05	2.03e-05	0.935	0.351	-2.11e-05	5.91e-05
x4	-1.623e-05	2.93e-05	-0.553	0.581	-7.41e-05	4.16e-05
x5	-0.2586	0.231	-1.120	0.264	-0.714	0.197
=====						
Omnibus:	51.669	Durbin-Watson:	2.044			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	38.997			
Skew:	-0.972	Prob(JB):	3.40e-09			
Kurtosis:	2.053	Cond. No.	3.18e+04			
=====						

Figura 4.9: Risultati ottenuti dal modello per il Community Smell SV

4.4.6 Modello di Regressione per Community Smell: SD

L'Immagine 4.11 riporta i dettagli relativi al modello statistico costruito per lo smell *Solution Defiance*. Il modello ha restituito come COVARIANCETYPE il valore "nonrobust", il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.362. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, tuttavia essendo leggermente maggiore indica che qualche variabile indipendente contribuisce alla variabilità della variabile dipendente (come conferma anche il valore di PROB(F-STATISTIC) pari a 0.0000000000000000182). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna P>|T|. I valori di questa colonna <0.05 sono quelli relativi a *Contributors* (x2) che corrisponde a 0.003, *CommitCount* (x3) che corrisponde a 0.000 e *BusFactorNumber* (x5) che corrisponde a 0.000. Da ciò si evince che *Contributors*, *CommitCount* e *BusFactorNumber* sono le variabili indipendenti che influenzano la presenza del Community Smell SD.

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.048			
Model:	OLS	Adj. R-squared:	0.024			
Method:	Least Squares	F-statistic:	1.966			
Date:	Tue, 26 Jul 2022	Prob (F-statistic):	0.0854			
Time:	16:22:37	Log-Likelihood:	-132.05			
No. Observations:	200	AIC:	276.1			
Df Residuals:	194	BIC:	295.9			
Df Model:	5					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	0.4590	0.262	1.751	0.082	-0.058	0.976
x1	-0.0017	0.012	-0.145	0.885	-0.025	0.022
x2	0.0088	0.004	2.151	0.033	0.001	0.017
x3	1.44e-05	2.16e-05	0.668	0.505	-2.81e-05	5.69e-05
x4	3.025e-06	3.11e-05	0.097	0.923	-5.84e-05	6.44e-05
x5	-0.4839	0.245	-1.974	0.050	-0.967	-0.001
Omnibus:	1171.674	Durbin-Watson:	1.899			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	28.932			
Skew:	0.584	Prob(JB):	5.22e-07			
Kurtosis:	1.547	Cond. No.	3.18e+04			

Figura 4.10: Risultati ottenuti dal modello per il Community Smell OS

4.4.7 Modello di Regressione per Community Smell: RS

L'Immagine 4.12 riporta i dettagli relativi al modello statistico costruito per lo smell *Radio Silence*. Il modello ha restituito come COVARIANCETYPE il valore "nonrobust", il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.109. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, tuttavia essendo leggermente maggiore indica che qualche variabile indipendente contribuisce alla variabilità della variabile dipendente (come conferma anche il valore di PROB(F-STATISTIC) pari a 0.000407). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna $P > |t|$. I valori di questa colonna < 0.05 sono quelli relativi a *Linguaggio di Programmazione* (x2) che corrisponde a 0.019, *CommitCount* (x3) che corrisponde a 0.017 e *BusFactorNumber* (x5) che corrisponde a 0.015. Da ciò si evince che *Linguaggio di Programmazione*, *CommitCount* e *BusFactorNumber* sono le variabili indipendenti che influenzano la presenza del Community Smell RS.

OLS Regression Results						
=====						
Dep. Variable:	y	R-squared:	0.362			
Model:	OLS	Adj. R-squared:	0.346			
Method:	Least Squares	F-statistic:	22.05			
Date:	Tue, 26 Jul 2022	Prob (F-statistic):	1.82e-17			
Time:	16:22:37	Log-Likelihood:	-40.579			
No. Observations:	200	AIC:	93.16			
Df Residuals:	194	BIC:	112.9			
Df Model:	5					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

const	-0.0454	0.166	-0.274	0.785	-0.373	0.282
x1	0.0111	0.008	1.467	0.144	-0.004	0.026
x2	-0.0077	0.003	-2.980	0.003	-0.013	-0.003
x3	-4.904e-05	1.36e-05	-3.593	0.000	-7.6e-05	-2.21e-05
x4	3.544e-05	1.97e-05	1.799	0.074	-3.41e-06	7.43e-05
x5	1.1580	0.155	7.465	0.000	0.852	1.464
=====						
Omnibus:	72.849	Durbin-Watson:	1.904			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	173.038			
Skew:	-1.668	Prob(JB):	2.66e-38			
Kurtosis:	6.104	Cond. No.	3.18e+04			
=====						

Figura 4.11: Risultati ottenuti dal modello per il Community Smell SD

4.4.8 Modello di Regressione per Community Smell: TFS

L'Immagine 4.13 riporta i dettagli relativi al modello statistico costruito per lo smell *Truck Factor Smell*. Il modello ha restituito come COVARIANCETYPE il valore "nonrobust", il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.036. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, tuttavia essendo leggermente maggiore indica che qualche variabile indipendente contribuisce alla variabilità della variabile dipendente (come conferma anche il valore di PROB(F-STATISTIC) pari a 0.205). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna P>|T|. L'unico valore di questa colonna <0.05 è quello relativo a *BusFactorNumber* (x5) che corrisponde a 0.029. Da ciò si evince che *BusFactorNumber* è l'unica variabile indipendente che influenza la presenza del Community Smell TFS.

OLS Regression Results						
=====						
Dep. Variable:	y	R-squared:	0.109			
Model:	OLS	Adj. R-squared:	0.086			
Method:	Least Squares	F-statistic:	4.749			
Date:	Tue, 26 Jul 2022	Prob (F-statistic):	0.000407			
Time:	16:22:37	Log-Likelihood:	-124.18			
No. Observations:	200	AIC:	260.4			
Df Residuals:	194	BIC:	280.1			
Df Model:	5					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

const	-0.6422	0.252	-2.549	0.012	-1.139	-0.145
x1	0.0273	0.011	2.373	0.019	0.005	0.050
x2	0.0029	0.004	0.742	0.459	-0.005	0.011
x3	5.002e-05	2.07e-05	2.413	0.017	9.14e-06	9.09e-05
x4	4.667e-05	2.99e-05	1.560	0.120	-1.23e-05	0.000
x5	0.5760	0.236	2.445	0.015	0.111	1.041
=====						
Omnibus:	465.149	Durbin-Watson:	2.035			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	25.108			
Skew:	0.512	Prob(JB):	3.53e-06			
Kurtosis:	1.598	Cond. No.	3.18e+04			
=====						

Figura 4.12: Risultati ottenuti dal modello per il Community Smell RS

4.4.9 Modello di Regressione per Community Smell: UI

L'Immagine 4.14 riporta i dettagli relativi al modello statistico costruito per lo smell *Unhealthy Interaction*. Il modello ha restituito come COVARIANCETYPE il valore "nonrobust", il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.023. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, tuttavia essendo leggermente maggiore indica che qualche variabile indipendente contribuisce alla variabilità della variabile dipendente (come conferma anche il valore di PROB(F-STATISTIC) pari a 0.0927). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna P>|T|. L'unico valore di questa colonna <0.05 è quello relativo a *Contributors* (x2) che corrisponde a 0.028. Da ciò si evince che *Contributors* è l'unica variabile indipendente che influenza la presenza del Community Smell UI.

4.4.10 Modello di Regressione per Community Smell: TC

L'Immagine 4.15 riporta i dettagli relativi al modello statistico costruito per lo smell *Toxic Communication*. Il modello ha restituito come COVARIANCETYPE il valore "nonrobust", il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.023. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, tuttavia essendo leggermente maggiore indica che qualche variabile indipendente contribuisce alla variabilità della variabile dipendente (come conferma anche il valore di PROB(F-STATISTIC) pari a 0.0927). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna P>|T|. L'unico valore di questa colonna <0.05 è quello relativo a *Contributors* (x2) che corrisponde a 0.028. Da ciò si evince che *Contributors* è l'unica variabile indipendente che influenza la presenza del Community Smell TC.

OLS Regression Results						
=====						
Dep. Variable:	y	R-squared:	0.036			
Model:	OLS	Adj. R-squared:	0.011			
Method:	Least Squares	F-statistic:	1.460			
Date:	Tue, 26 Jul 2022	Prob (F-statistic):	0.205			
Time:	16:22:37	Log-Likelihood:	-138.17			
No. Observations:	200	AIC:	288.3			
Df Residuals:	194	BIC:	308.1			
Df Model:	5					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

const	0.7319	0.270	2.709	0.007	0.199	1.265
x1	0.0089	0.012	0.726	0.469	-0.015	0.033
x2	0.0047	0.004	1.118	0.265	-0.004	0.013
x3	3.494e-06	2.22e-05	0.157	0.875	-4.04e-05	4.73e-05
x4	-1.302e-05	3.21e-05	-0.406	0.685	-7.63e-05	5.03e-05
x5	-0.5563	0.253	-2.202	0.029	-1.055	-0.058
=====						
Omnibus:	1436.124	Durbin-Watson:	2.143			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	30.161			
Skew:	0.382	Prob(JB):	2.82e-07			
Kurtosis:	1.258	Cond. No.	3.18e+04			
=====						

Figura 4.13: Risultati ottenuti dal modello per il Community Smell TFS

st'', il che significa che non c'è stata eliminazione di dati per calcolare la covarianza tra le caratteristiche. Il valore di R-SQUARED è 0.122. Quando questo valore è 0 significa che le variabili indipendenti date in input al modello non spiegano la variabilità, tuttavia essendo leggermente maggiore indica che qualche variabile indipendente contribuisce alla variabilità della variabile dipendente (come conferma anche il valore di PROB(F-STATISTIC) pari a 0.000116). Per individuare quale variabile indipendente influenza la presenza della variabile dipendente sono stati analizzati i valori ottenuti nella colonna $P > |t|$. L'unico valore di questa colonna < 0.05 è quello relativo a *CommitCount* (x3) che corrisponde a 0.000. Da ciò si evince che *CommitCount* è l'unica variabile indipendente che influenza la presenza del *Community Smell TC*.

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.047			
Model:	OLS	Adj. R-squared:	0.023			
Method:	Least Squares	F-statistic:	1.922			
Date:	Tue, 26 Jul 2022	Prob (F-statistic):	0.0927			
Time:	16:22:37	Log-Likelihood:	-138.35			
No. Observations:	200	AIC:	288.7			
Df Residuals:	194	BIC:	308.5			
Df Model:	5					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	0.4406	0.270	1.629	0.105	-0.093	0.974
x1	0.0131	0.012	1.065	0.288	-0.011	0.037
x2	-0.0093	0.004	-2.216	0.028	-0.018	-0.001
x3	4.042e-05	2.23e-05	1.817	0.071	-3.47e-06	8.43e-05
x4	3.293e-05	3.21e-05	1.025	0.306	-3.04e-05	9.63e-05
x5	0.1454	0.253	0.575	0.566	-0.353	0.644
Omnibus:	1313.626	Durbin-Watson:	1.964			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	28.988			
Skew:	0.241	Prob(JB):	5.07e-07			
Kurtosis:	1.198	Cond. No.	3.18e+04			

Figura 4.14: Risultati ottenuti dal modello per il Community Smell UI

RQ₄: Sommario dei Risultati.

La variabile *Linguaggio di Programmazione*, in generale, non risulta essere una variabile che influenza l'emergere dei Community Smells in una comunità di sviluppo, ad eccezione del Community Smell *Radio Silence (RS)*.


```

=====
                        OLS Regression Results
=====
Dep. Variable:          y      R-squared:          0.122
Model:                  OLS    Adj. R-squared:       0.099
Method:                 Least Squares    F-statistic:       5.387
Date:                  Tue, 26 Jul 2022    Prob (F-statistic): 0.000116
Time:                  16:22:37    Log-Likelihood:    -113.75
No. Observations:      200    AIC:              239.5
Df Residuals:          194    BIC:              259.3
Df Model:              5
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
const	0.1175	0.239	0.491	0.624	-0.354	0.589
x1	-0.0162	0.011	-1.489	0.138	-0.038	0.005
x2	0.0063	0.004	1.679	0.095	-0.001	0.014
x3	7.585e-05	1.97e-05	3.855	0.000	3.7e-05	0.000
x4	1.405e-05	2.84e-05	0.495	0.621	-4.2e-05	7.01e-05
x5	-0.1190	0.224	-0.532	0.595	-0.560	0.322

```

=====
Omnibus:                19.490    Durbin-Watson:          2.107
Prob(Omnibus):          0.000    Jarque-Bera (JB):       19.327
Skew:                   0.704    Prob(JB):               6.36e-05
Kurtosis:               2.418    Cond. No.:              3.18e+04
=====

```

Figura 4.15: Risultati ottenuti dal modello per il Community Smell TC

Osservazioni sulla Ricerca e sui Risultati

Questo capitolo illustra le osservazioni sulla ricerca, sui risultati ottenuti, le threats to validity dello studio e il modo in cui queste ultime sono state attenuate.

L'obiettivo dello studio, in particolare nelle domande di ricerca tre e quattro (**RQ₃** e **RQ₄**), è stato quello di comprendere il ruolo del linguaggio di programmazione principale utilizzato nello sviluppo di un progetto software come fattore che contribuisce all'insorgere di problemi di collaborazione e comunicazione, rappresentabili attraverso i Community Smells. Per ottenere i risultati dello studio è stato adottato un approccio di ricerca quantitativo, basato su modelli di Regressione Lineare Multipla. In particolare, la modellazione statistica ha indicato che il linguaggio di programmazione principalmente utilizzato non influenza il numero di Community Smells nelle comunità open-source, ad eccezione dello Smell *Radio Silence*.

5.1 Osservazioni

L'attuale stato dell'arte ha già studiato a fondo su come il linguaggio di programmazione influenzi la qualità del codice sorgente (come dettagliato nella sezione 2.2), ma non è stato definito se questa scarsa qualità sia frutto non solo del linguaggio di programmazione utilizzato ma bensì anche di problematiche di comunicazione e collaborazione all'interno del team. Nel presente lavoro di tesi, si è cercato di fare chiarezza su questo aspetto attraverso una ricerca empirica quantitativa.

5.1.1 Esecuzione del Tool CSDetector su repository (RQ₃)

A seguito dell'esecuzione del tool CSDetector sulle 200 repository del dataset, è stato possibile effettuare un primo studio statistico riguardo la presenza dei Community Smells per ogni linguaggio di programmazione. I risultati, come riportato nella Figura 4.5 e nella Tabella 4.2, hanno dimostrato che nel complesso i Community Smells maggiormente presenti nel dataset sono *Prima-Donnas Effect* (PDE) e *Solution Defiance* (SD).

Si ricorda che il Community Smell *Prima-Donnas Effect* compare quando un gruppo di sviluppatori non è disposto a rispettare i cambiamenti esterni degli altri membri del team a causa di una collaborazione strutturata in modo inefficiente all'interno di una comunità, mentre il Community Smell *Solution Defiance* si verifica quando la comunità di sviluppatori presenta diversi livelli di background culturale e di esperienza, e queste differenze portano alla divisione della comunità in sottogruppi simili con opinioni completamente contrastanti riguardo alle decisioni tecniche o socio-tecniche da prendere.

Partendo dalla definizione teorica appena enunciata era prevedibile un'alta presenza di questi Smells nelle repository del dataset per via della loro natura open-source.

È chiaro, infatti, che in repository open-source la collaborazione potrebbe non essere ben strutturata, causando la presenza dello Smell *PDE*, e alla stesura del codice potrebbero contribuire sviluppatori provenienti da parti del mondo opposte e con competenze/o conoscenze molto diverse tra loro, causando la presenza dello Smell *SD*.

Inoltre, i risultati hanno riportato una bassa percentuale di presenza per gli Smells *Organizational Silo Effect* (OSE), *Organizational Skirmish* (OS) e *Sharing Villainy* (SV) che sono causati rispettivamente da (1) presenza di sottogruppi isolati e mancanza di comunicazione e collaborazione tra gli sviluppatori della comunità, (2) disallineamento tra diversi livelli di competenza e canali di comunicazione tra unità di sviluppo o individui coinvolti nel progetto e (3) mancanza di attività di scambio di informazioni di alta qualità (ad es. incontri faccia a faccia). Questo risultato è andato contro le previsioni iniziali dato che le cause di presenza dei tre Smells elencati sono spesso verificate in repository open-source.

5.1.2 Modello Statistico (RQ₄)

Partendo dall'ipotesi che il linguaggio di programmazione influenza la qualità del codice sorgente e analizzando degli Smells prettamente legati a problemi di comunicazione e collaborazione nei team e nelle comunità di sviluppo e che influiscono sulla buona riuscita del prodotto finale, ci si aspettava che sulla presenza di questi Smells il linguaggio di pro-

grammazione influisse. I risultati ottenuti dal modello statistico sono andati contro le iniziali aspettative dimostrando che il linguaggio di programmazione, invece, non influisce sulla presenza dei Community Smells. L'unico Smell la cui presenza è influenzata dal linguaggio di programmazione è *Radio Silence*.

Si ricorda che il Community Smell *Radio Silence* si verifica quando si ha un'elevata formalità delle procedure regolari a causa dell'inefficiente organizzazione strutturale di una comunità. Il Community Smell RS causa tipicamente un ritardo nei cambiamenti e una perdita di tempo prezioso a causa di procedure formali complesse e rigide. L'effetto principale di questo Smell è un inatteso e massiccio ritardo nel processo decisionale a causa delle azioni formali necessarie.

Il risultato ottenuto è molto interessante poichè questo Smell, come si può intuire, riguarda l'organizzazione strutturale di una comunità di sviluppo. Lo studio è stato basato su comunità open-source e quindi è chiaro che la comunità potrebbe non essere strutturata in modo adeguato. Pertanto, per questo Smell in particolare, non ci si aspettava che il linguaggio di programmazione potesse influire. D'altra parte, ad esempio, ci si aspettava che il linguaggio di programmazione potesse influire sul Community Smell *Organizational Silo Effect* (che si ricorda essere causato dalla presenza di sottogruppi isolati di sviluppatori con conseguente assenza di comunicazione e collaborazione tra di essi), dato che una delle conseguenze è la presenza di codice duplicato che può essere favorita o sfavorita in base al linguaggio di programmazione utilizzato.

5.2 Threats to Validity

5.2.1 Threats to Construct Validity

Le threats to construct validity riguardano la relazione tra teoria e osservazione, e sono dovute principalmente all'imprecisione delle misurazioni effettuate.

Le informazioni tecniche riguardo le repository sono state estratte dal dataset *GH Archive* presente su GOOGLE BIGQUERY. I valori del dataset si aggiornano automaticamente ogni ora, pertanto risultano essere abbastanza accurati. Tuttavia, il numero di contributors per ogni repository non era esplicitamente riportato, ma è stato calcolato considerando il numero differente di account che hanno eseguito almeno un commit nella stessa. Naturalmente, non si possono escludere possibili imprecisioni nel calcolo di tale variabile, dato che un commit potrebbe essere eseguito anche con una sola modifica non rilevante, non contribuendo realmente alla stesura del codice sorgente.

Per fare in modo che un calcolo errato non influisse sul risultato finale, è stata calcolata approssimativamente una media del numero di contributors per tutte le repository del dataset e si è cercato di proseguire con le repository il cui numero di contributors si avvicinasse quanto più possibile a quello ottenuto.

5.2.2 Threats to Internal Validity

Le threats to internal validity riguardano fattori che potrebbero aver influenzato i risultati ottenuti senza che il ricercatore ne sia a conoscenza.

La maggior parte di queste minacce riguarda i modelli di regressione lineare multipla costruiti per l'ultima domanda di ricerca **RQ₄**. Poiché i fattori che influenzano la presenza di Community Smells sono un campo di ricerca ancora poco studiato, la possibilità di omettere alcune variabili nella costruzione del modello è eccezionalmente alta. Pertanto, una prima strategia di mitigazione è consistita nell'utilizzare tutti i fattori identificati da altri studi come correlati a Community Smells. Successivamente, è stata studiata l'influenza delle variabili di controllo costruendo modelli di regressione lineare utilizzati per valutare l'impatto dei fattori sulla variabile di risposta.

5.2.3 Threats to External Validity

Le threats to external validity sono condizioni che limitano la capacità di generalizzare i risultati dell'esperimento al mondo reale.

Per quanto riguarda la generalizzabilità dei risultati, dal punto di vista quantitativo è stata condotta un'analisi empirica su larga scala che ha coinvolto per le domande di ricerca 10 linguaggi di programmazione e 200 repository open-source con un numero di contributors compreso tra 30 e 50. Tuttavia, si ha la consapevolezza che l'elevato numero di errori generato dal tool **CSDetector** porta alla riduzione del set di dati utilizzabili per il raggiungimento dell'obiettivo principale. Pertanto, anche se i dati utilizzabili sono molti, sarà necessario eseguire un refactoring del suddetto tool per ampliare lo studio e consentire alla ricerca di progredire.

Questo capitolo illustra una sintesi della ricerca e delle prospettive future.

6.1 Conclusioni

Questa tesi riporta evidenze empiriche per chiarire la connessione tra *Linguaggio di Programmazione* e l'emergere di Community Smells (problemi di collaborazione e comunicazione) nelle comunità open-source.

L'obiettivo è stato analizzare e illustrare, attraverso l'uso di una strategia di ricerca quantitativa, se il linguaggio di programmazione utilizzato per la stesura di un codice sorgente influenzi l'emergere di problemi di collaborazione e comunicazione, rappresentati attraverso i Community Smells, all'interno dei team di sviluppo software open-source.

A partire dall'obiettivo principale, sono stati posti i seguenti sotto-obiettivi:

- Riportare informazioni sulle caratteristiche tecniche delle repository open-source di GITHUB.
- Riportare l'impatto dei linguaggi di programmazione sulla manifestazione dei Community Smells nelle comunità di sviluppo software.

Al fine di raggiungere i suddetti obiettivi, è stato eseguito uno studio quantitativo attraverso la costruzione di un modello di regressione lineare multipla statistico per mettere in relazione il Linguaggio di Programmazione con i dieci tipi di Community Smells, ossia

Organizational Silo Effect, Black-cloud Effect, Prima-donnas Effect, Sharing Villainy, Organizational Skirmish, Solution Defiance, Radio Silence, Truck Factor Smell, Unhealthy Interaction e Toxic Communication. Lo studio statistico svolto rivela che il Linguaggio di Programmazione principale in un progetto non influenza l'emergere di Community Smells, ad eccezione dello Smell *Radio Silence*.

In sintesi, il lavoro ha apportato i seguenti contributi:

1. Fornire informazioni riguardanti i linguaggi di programmazione maggiormente utilizzati nei progetti open-source di GITHUB, e il numero di contributors delle repository, attraverso uno studio di ricerca statistico.
2. Fornire informazioni riguardanti la relazione che esiste tra Community Smells e linguaggio di programmazione dominante all'interno del codice sorgente di un prodotto, attraverso la costruzione di un modello di regressione lineare multipla statistico.
3. Una repository GITHUB [20] per rendere disponibile pubblicamente dati, script e altro materiale utilizzato nel presente lavoro.

6.2 Lavori Futuri

In questa sezione verranno esposti eventuali lavori di ricerca futuri correlati al nostro.

Progetti privati Lo studio si è basato su progetti open-source ottenuti da GITHUB, una piattaforma open-source. È molto probabile che, per tale motivo, i risultati non rispecchino il mondo dei progetti privati (personali o progetti aziendali). Pertanto, un possibile studio futuro potrebbe riguardare proprio l'estensione del presente studio nel contesto dei progetti privati.

Classi di Linguaggi Lo studio si è basato sui 10 linguaggi di programmazione maggiormente utilizzati nelle repository GITHUB. Si è cercato di individuare 10 linguaggi con caratteristiche differenti tra loro. D'altra parte, sarebbe interessante ampliare lo studio andando a studiare se caratteristiche comuni a più linguaggi possono influenzare l'insorgere di Community Smells.

Ulteriori Fattori Tecnici Lo studio si è basato sul fattore tecnico "Linguaggio di Programmazione" e sull'eventuale correlazione tra questo fattore e l'insorgere di Community Smells.

Pertanto, un possibile lavoro futuro potrebbe riguardare lo studio di una eventuale influenza tra ulteriori fattori tecnici (ad esempio tool di comunicazione, ambienti di sviluppo, ecc.) e la presenza di Community Smells all'interno del codice sorgente.

Altri Community Smells Lo studio si è basato sui 10 Community Smells restituiti in output dal tool CSDETECTOR. Un possibile studio futuro potrebbe riguardare l'estensione dello studio ad ulteriori tipi di Community Smells, qualora venga implementato un tool capace di individuarne altri.

Ringraziamenti

Questo lavoro di tesi è la conclusione di un lungo percorso durato cinque anni, e vorrei ringraziare tutte persone che hanno contribuito al raggiungimento di questo personale obiettivo, e che con me hanno affrontato grandi difficoltà e gioito per le grandi soddisfazioni.

Ringrazio la professoressa Filomena Ferrucci, relatrice per questo lavoro di tesi. Non ho avuto il piacere di condividere con lei le soddisfazioni della laurea triennale, ma durante la laurea magistrale si è sempre dimostrata una persona competente e disponibile, appassionata a ciò che insegna. Grazie a lei ho avuto la certezza di aver intrapreso la strada giusta per questo mio percorso accademico.

Ringrazio Stefano Lambiase, mio supervisore durante questo lavoro di tesi. In lui ho trovato una persona cordiale, competente e sempre disponibile. Grazie di avermi accompagnato in questo passo finale del mio percorso.

Ringrazio tutti i professori incontrati nel mio percorso di studi, in particolare il professore Andrea De Lucia che grazie alla sua competenza e alla passione che dimostra ogni giorno nel suo lavoro mi ha fatto innamorare dell'ingegneria del software. Se ho intrapreso questo percorso è anche grazie a lui.

Ringrazio tutta la mia famiglia, che mi è stata vicino in ogni momento bello e brutto e che mi ha sempre incoraggiata a fare di meglio. Non avete mai smesso di credere in me, e spero di rendervi ogni giorno fieri dei miei successi e della donna che sono diventata. Se sono quello che sono è soprattutto grazie a voi.

Ringrazio Salvatore, che nonostante alti e bassi che capitano in una coppia non ha mai

smesso di dimostrarmi quanto io valga. Grazie per non avermi lasciata mai sola e per avermi dimostrato ogni giorno di tenerci.

Ringrazio i miei colleghi universitari, la pandemia e la distanza non ci hanno mai realmente separato. Grazie per esserci stati sempre.

Ringrazio i miei colleghi di lavoro, nonostante non ci siamo mai visti di persona, tra una chiacchiera e l'altra avete reso un po' più leggere le mie giornate, permettendomi di riuscire a conciliare lavoro e studio senza troppe difficoltà. Grazie per avermi dimostrato di credere in me in ogni caso.

Infine, ringrazio me stessa per la mia grande forza, per la mia testardaggine, per la mia continua voglia di imparare, per la mia ambiziosità, per la mia determinazione. Tutte caratteristiche che mi hanno portato ad arrivare fin qui e ad essere la persona che sono, spero di non cambiare mai.

- [1] Narciso Cerpa e June M. Verner. "Why Did Your Project Fail?" In: *Commun. ACM* 52.12 (dic. 2009), pp. 130–134. ISSN: 0001-0782. DOI: 10.1145/1610252.1610286. URL: <https://doi.org/10.1145/1610252.1610286>.
- [2] Magne Jørgensen e Kjetil Moløkken-Østvold. "How large are software cost overruns? A review of the 1994 CHAOS report". In: *Information and Software Technology* 48.4 (2006), pp. 297–301.
- [3] Jim Johnson. "CHAOS 2020: Beyond Infinity". In: *Standish Group* (2020).
- [4] Taimour Al Neimat. "Why IT projects fail". In: *The project perfect white paper collection* 8 (2005).
- [5] Terry Frieden. "Report: FBI wasted millions on 'Virtual Case File'". In: *Retrieved April 9* (2005), p. 2005.
- [6] Azham Hussain e Emmanuel OC Mkpojiogu. "Requirements: Towards an understanding on why software projects fail". In: *AIP Conference Proceedings*. Vol. 1761. 1. AIP Publishing LLC. 2016, p. 020046.
- [7] Project Management Institute. *A Guide to the Project Management Body of Knowledge*. 7^a ed. Ago. 2021, p. 250. ISBN: 1628256648.
- [8] Frederick P Brooks Jr. *The mythical man-month: essays on software engineering*. Pearson Education, 1995.

- [9] Paul Ralph, Mike Chiasson e Helen Kelley. "Social Theory for Software Engineering Research". In: *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. EASE '16. Limerick, Ireland: Association for Computing Machinery, 2016. ISBN: 9781450336918. DOI: 10.1145/2915970.2915998. URL: <https://doi.org/10.1145/2915970.2915998>.
- [10] Sébastien Cherry e Pierre N Robillard. "Communication problems in global software development: Spotlight on a new field of investigation". In: *International Workshop on Global Software Development, International Conference on Software Engineering, Edinburgh, Scotland*. IET. 2004, pp. 48–52.
- [11] Sarah Beecham, Padraig OLeary, Ita Richardson, Sean Baker e John Noll. "Who are we doing global software engineering research for?" In: *2013 IEEE 8th International Conference on Global Software Engineering*. IEEE. 2013, pp. 41–50.
- [12] Gemma Catolino, Fabio Palomba, Damian A Tamburri, Alexander Serebrenik e Filomena Ferrucci. "Gender diversity and women in software teams: How do they affect community smells?" In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE. 2019, pp. 11–20.
- [13] Stefano Lambiase, Gemma Catolino, Damian A. Tamburri, Fabio Palomba, Filomena Ferrucci e Alexander Serebrenik. "Good Fences Make Good Neighbours? On the Impact of Cultural and Geographical Dispersion on Community Smells". In: (2022), p. 12. DOI: 10.1145/3510458.3513015.
- [14] Fabio Palomba, Damian Andrew Tamburri, Francesca Arcelli Fontana, Rocco Oliveto, Andy Zaidman e Alexander Serebrenik. "Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?" In: *IEEE Transactions on Software Engineering* 47.1 (2021), pp. 108–129. DOI: 10.1109/TSE.2018.2883603.
- [15] Damian A Tamburri, Rick Kazman e Hamed Fahimi. "The architect's role in community shepherding". In: *IEEE Software* 33.6 (2016), pp. 70–79.
- [16] Damian A. Tamburri, Fabio Palomba e Rick Kazman. "Exploring Community Smells in Open-Source: An Automated Approach". In: *IEEE Transactions on Software Engineering* 47.3 (2021), pp. 630–652. DOI: 10.1109/TSE.2019.2901490.
- [17] Damian A. Tamburri. "Software Architecture Social Debt: Managing the Incommunicability Factor". In: *IEEE Transactions on Computational Social Systems* 6.1 (2019), pp. 20–37. DOI: 10.1109/TCSS.2018.2886433.

- [18] Nuri Almarimi, Ali Ouni e Mohamed Wiem Mkaouer. "Learning to detect community smells in open source software projects". In: *Knowledge-Based Systems* 204 (2020), p. 106201.
- [19] Baishakhi Ray, Daryl Posnett, Vladimir Filkov e Premkumar Devanbu. "A Large Scale Study of Programming Languages and Code Quality in Github". In: *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. Vol. 1. FSE. 2014, pp. 155–165.
- [20] Vidoni. *Impatto dei linguaggi di programmazione sulla presenza di Community Smells: uno studio empirico*. 2022. URL: <https://github.com/AliceVidoni/LanguageProgramming-for-CommunitySmells>.
- [21] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Fausto Fasano, Rocco Oliveto e Andrea De Lucia. "On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation". In: *Empirical Software Engineering* 23.3 (2018), pp. 1188–1221.
- [22] Michele Tufano, Fabio Palomba, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Andrea De Lucia e Denys Poshyvanyk. "When and why your code starts to smell bad (and whether the smells go away)". In: *IEEE Transactions on Software Engineering* 43.11 (2017), pp. 1063–1088.
- [23] Naouel Moha, Yann-Gaël Guéhéneuc, Laurence Duchien e Anne-Francoise Le Meur. "Decor: A method for the specification and detection of code and design smells". In: *IEEE Transactions on Software Engineering* 36.1 (2009), pp. 20–36.
- [24] Fabio Palomba, Annibale Panichella, Andrea De Lucia, Rocco Oliveto e Andy Zaidman. "A textual-based technique for smell detection". In: *2016 IEEE 24th international conference on program comprehension (ICPC)*. IEEE. 2016, pp. 1–10.
- [25] Stefano Lambiase, Andrea Cupito, Fabiano Pecorelli, Andrea De Lucia e Fabio Palomba. "Just-In-Time Test Smell Detection and Refactoring: The DARTS Project". In: *Proceedings of the 28th International Conference on Program Comprehension*. 2020, pp. 441–445.
- [26] Ward Cunningham. "The WyCash portfolio management system". In: *ACM SIGPLAN OOPS Messenger* 4.2 (1992), pp. 29–30.
- [27] Melvin E Conway. "How do committees invent". In: *Datamation* 14.4 (1968), pp. 28–31.

- [28] D.A. Tamburri, P. Kruchten, P. Lago e H. van Vliet. "Social Debt in Software Engineering: Insights from Industry". English. In: *Journal of Internet Services and Applications* (2015). ISSN: 1867-4828. DOI: 10.1186/s13174-015-0024-6.
- [29] Damian A Tamburri, Philippe Kruchten, Patricia Lago e Hans van Vliet. "What is social debt in software engineering?" In: *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE. 2013, pp. 93–96.
- [30] Damian Andrew Andrew Tamburri, Fabio Palomba e Rick Kazman. "Exploring community smells in open-source: An automated approach". In: *IEEE Transactions on software Engineering* (2019).
- [31] Mitchell Joblin, Wolfgang Mauerer, Sven Apel, Janet Siegmund e Dirk Riehle. "From developer networks to verified communities: A fine-grained approach". In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. IEEE. 2015, pp. 563–573.
- [32] Michele Tufano, Fabio Palomba, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Andrea De Lucia e Denys Poshyvanyk. "When and why your code starts to smell bad". In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. IEEE. 2015, pp. 403–414.
- [33] Nuri Almarimi, Ali Ouni, Moataz Chouchen, Islem Saidani e Mohamed Wiem Mkaouer. "On the detection of community smells using genetic programming-based ensemble classifier chain". In: *Proceedings of the 15th International Conference on Global Software Engineering*. 2020, pp. 43–54.
- [34] Fabio Palomba e Damian Andrew Tamburri. "Predicting the emergence of community smells using socio-technical metrics: a machine-learning approach". In: *Journal of Systems and Software* 171 (2021), p. 110847.
- [35] Gemma Catolino, Fabio Palomba, Damian A Tamburri, Alexander Serebrenik e Filomena Ferrucci. "Refactoring community smells in the wild: the practitioner's field manual". In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*. 2020, pp. 25–34.
- [36] Antonio Martini e Jan Bosch. "Revealing social debt with the CAFFEA framework: An antidote to architectural debt". In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE. 2017, pp. 179–181.

- [37] Damian A Tamburri, Fabio Palomba, Alexander Serebrenik e Andy Zaidman. "Discovering community patterns in open-source: a systematic approach and its evaluation". In: *Empirical Software Engineering* 24.3 (2019), pp. 1369–1417.
- [38] Guilherme Avelino, Leonardo Passos, Andre Hora e Marco Tulio Valente. "A novel approach for estimating truck factors". In: *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*. IEEE. 2016, pp. 1–10.
- [39] Damian A Tamburri e Elisabetta Di Nitto. "When software architecture leads to social debt". In: *2015 12th Working IEEE/IFIP Conference on Software Architecture*. IEEE. 2015, pp. 61–64.
- [40] Helen Sharp e Hugh Robinson. "Some social factors of software engineering: the maverick, community and technical practices". In: *Proceedings of the 2005 workshop on Human and social factors of software engineering*. 2005, pp. 1–6.
- [41] S Magnoni, DA Tamburri, E Di Nitto e R Kazman. "Analyzing quality models for software communities". In: *Communications of the ACM-: Under Review* (2017).
- [42] Marcelo Cataldo, James D Herbsleb e Kathleen M Carley. "Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity". In: *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. 2008, pp. 2–11.
- [43] Marcelo Cataldo, Patrick A Wagstrom, James D Herbsleb e Kathleen M Carley. "Identification of coordination requirements: Implications for the design of collaboration and awareness tools". In: *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. 2006, pp. 353–362.
- [44] Giuseppe Valetto, Sunita Chulani e Clay Williams. "Balancing the value and risk of socio-technical congruence". In: *Workshop on Sociotechnical Congruence*. Vol. 4. Citeseer. 2008.
- [45] Javier Portillo-Rodríguez, Aurora Vizcaino, Mario Piattini e Sarah Beecham. "Using agents to manage socio-technical congruence in a global software engineering project". In: *Information Sciences* 264 (2014), pp. 230–259.
- [46] Melvin E Conway. "How do committees invent". In: *Datamation* 14.4 (1968), pp. 28–31.
- [47] Irwin Kwan, Adrian Schroter e Daniela Damian. "Does socio-technical congruence have an effect on software build success? a study of coordination in a software project". In: *IEEE Transactions on Software Engineering* 37.3 (2011), pp. 307–324.

- [48] Erik Trainer, Stephen Quirk, Cleidson de Souza e David Redmiles. "Bridging the gap between technical and social dependencies with ariadne". In: *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*. 2005, pp. 26–30.
- [49] Anita Sarma, Larry Maccherone, Patrick Wagstrom e James Herbsleb. "Tesseract: Interactive visual exploration of socio-technical relationships in software development". In: *2009 IEEE 31st International Conference on Software Engineering*. IEEE. 2009, pp. 23–33.
- [50] Fabio Palomba, Damian A Tamburri, Alexander Serebrenik, Andy Zaidman, Francesca Arcelli Fontana e Rocco Oliveto. "Poster: How Do Community Smells Influence Code Smells?" In: *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*. IEEE. 2018, pp. 240–241.
- [51] Laurie Williams e Robert R Kessler. *Pair programming illuminated*. Addison-Wesley Professional, 2003.
- [52] Mívia Ferreira, Marco Tulio Valente e Kecia Ferreira. "A comparison of three algorithms for computing truck factors". In: *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*. IEEE. 2017, pp. 207–217.
- [53] Skipper Seabold e Josef Perktold. "Statsmodels: Econometric and statistical modeling with python". In: *9th Python in Science Conference*. 2010.