Empirical Software Engineering (2025) 30:130
https://doi.org/10.1007/s10664-025-10673-7

Does microservice adoption impact the velocity? A cohort study

Nyyti Saarimäki[1] · Mikel Robredo[2] · Valentina Lenarduzzi[2] · Sira Vegas[3] · Natalia Juristo[3] · Davide Taibi[2]

Accepted: 5 May 2025 / Published online: 25 June 2025

Abstract
[Context] Microservices enable the decomposition of applications into small, independent, and connected services. The independence between services could positively affect a project's velocity, which is considered an important maintenance metric measuring the time taken to implement features and fix bugs. However, no studies have investigated the causal relationship between microservices and velocity. [Objective and Method] The goal of this study is to investigate the effect of microservices on velocity which is a common maintenance metric. The study compares projects on GitHub developed with microservices style from the beginning and similar projects using monolithic architectures. The study was conducted as a retrospective cohort study, which is a study type used to assess causality. [Results] The results did not find statistically significant differences in mean velocities in microservice-based and monolithic projects. Furthermore, the statistical adjustment performed to quantify the statistical impact of the use of microservices on velocity considering additional confounders did not find statistically significant impact from these. [Conclusions] The results did not indicate a difference between microservices-based projects and monolithic projects in terms of velocity. In addition, this study will contribute to the body of knowledge of empirical methods and be among the first works to adopt the methodology of the cohort study.

Keywords Microservices · Velocity · Cohort study · Empirical software validation · Mining software repositories

1 Introduction

Microservices have become popular in the last few years, as they are considered to have several benefits over the traditional monolithic structure (non-MS). They divide a project into smaller autonomous services, each with a clear purpose (Fowler and Lewis 2014). Therefore, the microservice architecture (MS) has a significant effect on the overall development of a project, allowing independent and concurrent development and deployment(Waseem et al. 2020),

Communicated by: Neil Ernst

This article is part of the Topical Collection: *Special Issue on Registered Reports*

Extended author information available on the last page of the article

reusability (Fowler and Lewis 2014), and improved scalability (Dragoni et al. 2018). Due to their independence in deployment, the adoption of MS should increase the velocity (Jangla 2018a). Velocity is an important maintenance measure in current software projects, and higher velocity implies faster implementation of features and faster resolution of issues. However, no studies investigate the connection between MS and velocity.

MS introduce various complexities into project development, including challenges inherent to distributed systems such as inter-service communication, the use of communication channels (e.g., message buses), and distributed tracing. These additional components often lead to increased development costs due to the need to manage these extra moving parts (Taibi et al. 2017; Soldani et al. 2018). Recently, several companies have reverted to monolithic systems, citing the increasing costs and higher complexity associated with MS. Therefore, despite the widespread belief among developers that MS may enhance development velocity, it is crucial to empirically evaluate this claim. It is important to determine whether any observed improvements in velocity are truly attributable to MS or if other confounding factors are at play.

Previous research, including several surveys and empirical studies, have investigated developers' perceptions of enhanced velocity with MS adoption (Taibi et al. 2017; Soldani et al. 2018; Wu et al. 2022; Lenarduzzi et al. 2020; Ayas et al. 2023; Lomio et al. 2022). However, none of these studies 1) checked the results achieved by analyzing projects that started their development with MS architectures, and 2) proved that MS causes improved development outcomes. This important gap shows that we need real evidence to confirm whether these improvements can be attributed to MS architecture, or whether they are influenced by other factors.

Controlled experiments are commonly adopted for assessing causality in Empirical Software Engineering and its validity has also been demonstrated in the past in different fields. However, sometimes running a controlled experiment is difficult because recruiting a suitable study sample for the study is not possible or could take a lot of resources (Saarimäki et al. 2020). Moreover, in other cases, when researchers have an existing dataset that contains a lot of data considered useful for validating their hypothesis, it is not possible to randomly allocate experimental units to treatments or to manipulate the treatments to be studied, as the data have already been collected when the investigation begins. Therefore, the only possible solution is to opt for correlation studies (Saarimäki et al. 2020).

To solve this problem and to make the most of the potential of a rich dataset, researchers in epidemiology have for years successfully adopted cohort studies (a type of observational study) (Di Cicco et al. 2016). Recent work has shown comparable results for cohort studies and controlled experiments (Benson and Hartz 2000; Concato et al. 2000).

In the MS context, the application of cohort studies finds fertile ground. Due to the data available from these various studies, a cohort study methodology presents an opportunity to apply a new analytical approach to this extensive data set. This methodology allows for a robust exploration of causal relationships by comparing actively developed projects that adopted MS from their inception against similar projects that use monolithic architectures (Gordis 2014; Grimes and Schulz 2002b; Morabia 2004). The objective of this approach is twofold: 1) to provide a corroboration of existing research and 2) to offer conclusive evidence on the impact of MS architecture on software development speed through a systematic analysis of confounding factors.

Therefore, we systematically investigate the causal effects of MS on development velocity, corroborating existing research by applying the cohort study methodology.

We designed and conducted a cohort study that compared actively developed projects that adopted MS from the beginning and similar projects that used a monolithic architecture. The proposed study plan covers all the aspects required for conducting the study, including data sources, variable definitions, and data analysis. As a result, the results of the study are expected to provide high-level evidence on the topic. Moreover, cohort studies include an important step in systematic analysis of confounding factors that will help us understand if the improved velocity (if any) will be due to the introduction of MS or not.

Therefore, the contribution of this paper includes the following:

- A cohort study investigating the impact of MS on development velocity, based on the investigation of a total of 265 open-source projects, where 58 were MS-based projects and 207 were non-MS-based projects.
- One of the first examples of cohort study application in empirical software engineering
- A step-by-step methodology description that will help researchers to conduct cohort studies.

The remainder of this paper is structured as follows. Section 2 describes the background and Section 3 the related work. Section 4 describes the design and execution of our study. Section 5 presents the results of the study, while Section 6 discusses them. Section 7 discusses the possible threats. Section 8 draws conclusions and plans for future work.

2 Background

In this Section, we describe the background of this work.

2.1 Microservices

The microservice style is derived from a Service-Oriented Architecture, where services have dedicated responsibilities, but are not independent (Fowler and Lewis 2014; Dragoni et al. 2017). In SOA, individual services are neither full-stack (e.g., the same database is shared among multiple services) nor fully autonomous (e.g., service A depends on service B). In contrast, MS systems are decentralized systems composed of a large number of small independent services that communicate through different lightweight mechanisms (Pahl et al. 2018).

MS are relatively small and autonomous services that are deployed independently, with a single and clearly defined purpose (Fowler and Lewis 2014). MS enables vertically decomposing applications into a subset of business-driven independent services. Each service can be developed, deployed, and tested independently by different development teams using different technologies stacks. MS has a variety of advantages (Jamshidi et al. 2018). They can be developed in different programming languages, scale independently from other services, and be deployed on the hardware that best suits their needs. Moreover, because of their size, they are easier to maintain and more fault tolerant, since the failure of one service will not disrupt the entire system, which could happen in a monolithic system (Fowler and Lewis 2014).

One of the major advantages claimed by practitioners is the independent deployment and, therefore, the increased deployment frequency and velocity of microservice-based systems. Although this claim is often mentioned as a theoretical advantage, to the best of our knowledge, no study investigates if it is true. The "2022 DORA Accelerate State of DevOps Report"[1] recommends the highest velocity possible for software companies, suggesting that startups

[1] https://dora.dev/

or newly born companies have a deployment frequency between once a day and once a week, and multiple deployments per hour to "flowing" and more mature companies, considering retiring companies that deploy between once per month and once every six months.

2.2 Docker

Docker[2] is a platform for developing, shipping and running applications in loosely isolated environments called containers. A Docker container includes everything needed to run an application, and, therefore, it enables the developer to develop on their platform of choice without having to worry about where the program is deployed (Jangla 2018b). This is one of the several reasons why Docker is a popular tool among developers and was voted the most important tool in StackOverflow's developer survey in 2022[3]. The tool is commercial, but it has a free version for open-source communities and individual developers. A Dockerized project can consist of one or several containers. The containers can communicate with each other, but the implementation of each container is otherwise independent of other containers. Therefore, Docker is one way to create MS. In practice, each container has a file `Dockerfile`. It is a "text document that contains all the commands a user could call on the command line to assemble an image." (Inc. 2023).

2.3 Velocity

Velocity is a maintenance metric that estimates the amount of productive work a developer team can complete in a given time frame. The influence of MS on velocity can vary depending on different factors, including

- *Independent Development and Deployment:* MS facilitate the autonomous development and deployment of individual services. This enables teams to work concurrently on different services, reducing dependencies and bottlenecks. Consequently, development cycles can be expedited, allowing quick iteration and delivery of new features or improvements (Soldani et al. 2018; Fowler and Lewis 2014).
- *Enhanced Scalability:* MS architecture empowers the independent scaling of individual services according to their specific requirements. This flexibility enables teams to optimize performance and responsiveness, ensuring efficient utilization of resources. Fine-grained scaling aligned with demand can increase velocity by effectively handling the increased workload (Dragoni et al. 2018).
- *Concurrent Development and Testing:* MS architecture facilitates parallel development and testing, as services can be developed and tested in isolation. Teams can independently work on different services, enabling concurrent progress. This significantly speeds up the development lifecycle, as changes and updates can be implemented in parallel, reducing the overall development time (Waseem et al. 2020).
- *Reusability and Modularity:* MS encourage the creation of small, reusable components that can be shared between services. This promotes reusability and modularity, accelerating development by leveraging existing services, libraries, and frameworks. Developers can build on existing functionalities, reduce redundant efforts, and speed up the development process (Fowler and Lewis 2014).

[2] Docker: https://www.docker.com/

[3] https://survey.stackoverflow.co/2022/#most-popular-technologies-tools-tech-prof

Ultimately, the influence of MS on velocity depends on the effectiveness of architecture design, the maturity level of microservice adoption, the skills and experience of the development teams, and the efficiency of the support processes and tools (Soldani et al. 2018; Taibi et al. 2017).

To our knowledge, this is the first study empirically investigating the impact of MS on velocity.

2.4 Cohort Study

Experiments are the golden standard for investigating causality. Their ability to provide high-level evidence is based on measuring an association between cause and outcome, ensuring the cause preceded the outcome, and controlling subjects and the environment (Oppewal 2010; Rosenbaum 2002). However, conducting experiments is not always possible or sensible, as the study could be unethical or require too much resources (e.g. time or money). Furthermore, experiments are not compatible with historical or observational data, which is especially problematic for mining software repository studies. A branch of medicine called epidemiology has developed retrospective cohort studies and other types of analytical observational study to address these issues. Methods have been used, for example, to prove that smoking causes lung cancer (Trichopoulos et al. 1981).

A cohort study (Grimes and Schulz 2002a; Gordis 2014) investigates whether exposure (independent variable) causes an outcome (dependent variable) by comparing the results of two or more groups of study subjects with different levels of exposure. Study subjects are *observed* over a predefined period to see what happens naturally. The observations can be made prospectively in real time or retrospectively from historical data. The general design of a continuous outcome cohort study is presented in Fig. 1.

The cohort studies select the study population from a source population (source of data) using unambiguous selection criteria. Their purpose is to ensure that all subjects are similar in important aspects. This is required for the validity of the study as the groups need to be comparable to answer the research question within the context of the population of interest.

After selecting the population, exposure, confounders, and outcome are measured for all study subjects at the beginning of follow-up. The outcome is measured at the end of the follow-up period. The data collected are then used to analyze whether there is a relationship between exposure and outcome. Note that when the outcome is continuous, it has to be

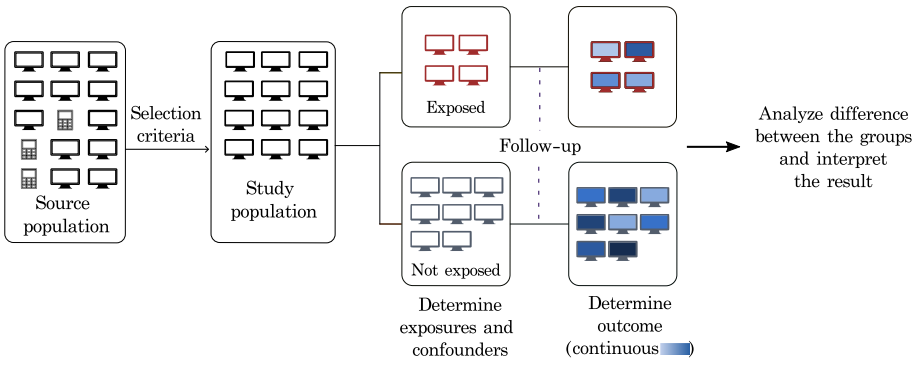

Fig. 1 The general study design of a cohort study with a continuous outcome

measured also at the start of the study to control for the starting point. This provides a temporal framework that makes it possible to assess causality.

Cohort studies are the study type considered to provide the highest level of evidence after experiments (Song and Chung 2010). Like them, cohort studies are capable of assessing the statistical association between variables and ensuring temporal precedence. However, the main difference in the level of evidence provided comes from controlling the environment. As the methodology is observational, the study has no control and the results can also be influenced by external factors such as project age or size. Therefore, in cohort studies, it is crucial to identify these factors from the literature or using domain knowledge, measure them, and control their effect.

3 Related Work

The adoption of MS is often overrated. Companies adopt them for various reasons: to enable larger teams to work on the same project by decomposing it into smaller, independent services, improving system scalability at runtime, or reducing costs (Taibi et al. 2017).

Unfortunately, not all companies experience the expected benefits and often encounter unforeseen problems or drawbacks. Examples include the complexity of decoupling from monolithic systems and increased development costs due to the higher number of components that need maintenance, such as Kubernetes, message buses, and API gateways (Soldani et al. 2018).

Different empirically investigated the benefits of MS. In particular, Lenarduzzi et al. (2020) performed a case study in a small and medium enterprises (SME) that decomposed their monolithic system into five MS, showing that extraction of a feature of the monolithic system as a microservice, after a stabilization time, has a lower amount of Technical Debt measured by SonarQube compared to the trend of the monolithic system. Their findings showed that extracting a feature from the monolithic system into a microservice, after a stabilization period, resulted in a lower amount of Technical Debt, as measured by SonarQube,[4] compared to the trend observed in the monolithic system.

The benefits of MS have been investigated through company surveys and empirical studies on existing projects. Lomio et al. (2022) conducted a exploratory correlational study to examine the trends in Accelerate Metrics before and after the adoption of MS within a large company. Their analysis observed a decrease in Lead Time for Changes (the average time taken for a commit to progress from deployment to production) and an increase in Deployment Frequency, which they attributed to improved team independence. However, the study did not aim to establish or claim a causal relationship between MS adoption and these metrics but rather identified correlations in historical data.

Ayas et al. (2023) conducted a mixed-methods study to investigate the migration process from monolithic to MS architectures. They performed a set of interviews with practitioners to understand how practitioners perceive and experience their migration journeys, identifying key activities and two "modes of change": systemic (strategic and organizational) and technical (focused on implementation). Moreover, they analyzed StackOverflow posts to validate the insights gathered from the interviews. Their study provides a comprehensive overview of the MS migration process but does not establish or claim causal relationships between migration decisions and development outcomes..

[4] SonarQube: https://www.sonarqube.org

Wu et al. (2022) study the types of problems that occur in MS systems, their causes, and the solutions implemented to address them. They analyzed data from open source MS projects on GitHub, interviewed 15, and conducted a survey of 150 MS practitioners. Also, in this case, developers' interviews confirm that MS can enhance development velocity even if they introduce specific challenges that need to be managed effectively. Also in this case, the authors didn't aim at establishing cause-effect relationships.

Our result will complement the existing body of knowledge by performing a large-scale analysis of a total of 265 open-source projects, of which 58 were MS-based projects and 207 were not, thus confirming whether the MS adoption can increase the velocity or not, using the cohort methods.

4 The Cohort Study Design

In this Section, we describe our empirical study design. All the following steps are described according to the STrengthening the Reporting of OBservational studies in Epidemiology (STROBE) guidelines (Von Elm et al. 2007) (Table 1).

4.1 Goal and Research Questions

The purpose of the study was to understand the effect of MS on velocity during the early stages of the evolution of a software project.

Therefore, our goal can be addressed by the following research question:

RQ *Do projects developed with a MS architecture close issues at a higher rate than projects developed with a monolithic architecture?*

The **hypothesis** was that GitHub projects using a microservice architectural style close issues at a faster rate than projects not using a microservice architecture. This outcome was expected because the decomposition of software projects into independent MS allows each service to be deployed and optimized according to their needs, which should accelerate the development process (Newman 2015; Jangla 2018a). Despite MS being a popular architectural style, there are no studies on the topic exploring its effect on velocity. Therefore, this is the first study to empirically investigate the impact of MS on velocity.

4.2 Study Design

The nature of the research question in this paper is understanding whether an independent variable affects a dependent variable. The most reliable method for investigating such an effect is a controlled experiment, and it is the only method considered to be able to assess causality. However, recruiting a sufficiently large set of professional developers to study the research question would have required significantly more resources than we had available.

The study was designed as a *retrospective cohort study*. We adopted it because it can utilize historical observational data on the evolution of software projects that are already available online. Although it provides a lower level of evidence than a controlled experiment, when conducted correctly, it provides more reliable results than a correlation study. The study follows over time and compares a set of young open-source software projects using the MS architecture with similar projects using the monolithic architecture.

Table 1 The items in STROBE Checklist describing the essential items to be reported in the method and results section of a cohort study

Method Section	Step Description
Study design	Present key elements of study design early in the paper.
Setting	Describe the setting, locations, and relevant dates, including periods of recruitment, exposure, follow-up, and data collection.
Subjects	(a) Give the eligibility criteria, and the sources and methods of selection of participants. Describe methods of follow-up.
	(b) For matched studies, give matching criteria and number of exposed and unexposed.
Variables	Clearly define all outcomes, exposures, predictors, potential confounders, and effect modifiers.
*Data sources & measurement**	For each variable of interest, give sources of data and details of methods of assessment (measurement). Describe the comparability of assessment methods if there is more than one group.
Study size	Explain how the study size was arrived at.
Statistical methods	(a) Describe all statistical methods, including those used to control for confounding.
	(b) Describe any methods used to examine subgroups and interactions.
	(c) Explain how missing data were addressed.
	(d) If applicable, explain how the loss to follow-up was addressed.
	(e) Describe any sensitivity analyses.
Results Section	Step Description
*Participants**	(a) Report numbers of individuals at each stage of study (eg numbers potentially eligible, examined for eligibility, confirmed eligible, included in the study, completing follow-up, and analyzed)
	(b) Give reasons for non-participation at each stage
	(c) Consider the use of a flow diagram
*Descriptive data**	(a) Give characteristics of study participants (eg demographic, clinical, social) and information on exposures and potential confounders
	(b) Indicate the number of participants with missing data for each variable of interest
	(c) Summarise follow-up time (eg, average and total amount)
Main results	(a) Give unadjusted estimates and, if applicable, confounder-adjusted estimates and their precision (eg, 95% confidence interval). Make clear which confounders were adjusted for and why they were included
	(b) Report category boundaries when continuous variables were categorized
	(c) If relevant, consider translating estimates of relative risk into absolute risk for a meaningful time period

* Give information separately for exposed and unexposed groups.
Please note that since our study did not involve humans. We renamed the step called "*Participants*" as "*Subjects*" and skipped irrelevant steps.
Von Elm et al. (2007)

4.3 Setting

The study investigated open-source projects on GitHub created during 1/2016-6/2021. The *cases* of the study were projects that had adopted a microservice architecture within 12 months from their creation date, while the *controls* of the study had a monolithic structure throughout the study. The overall design is depicted in Fig. 2.

The projects were tracked for 12 months, starting from when they were 12 months old until they reached two years of age. Our aim was to analyze young but consistently evolving software projects. We decided to start observing projects after their first year of development as during the first 12 months of MS projects evolution they are still evolving quickly (Lenarduzzi et al. 2020; Newman 2021). Therefore, monitoring MS projects during the first 12 months of life would not allow us to study stable projects during their growth phase. The length of the follow-up was selected to be 12 months as that time was considered to show differences but not provide enough time for the project evolve into a completely different project by the end of the follow-up.

Therefore, study subjects had different data measurement dates, but the same fixed duration of follow-up time (Fig. 2). The registered report proposed an 18-month project creation window between 1/2020 and 6/2021 (Saarimäki et al. 2023). The window was selected based on an initial data crawl from the World of Code database. However, we decided against directly using World of Code as we found a data set from Amoroso d'Aragona et al. (2024) containing a curated list of docker-based projects. The data set had only 40 projects created during the initial defined window, and therefore we decided to extend the project creation window. Increasing the project creation window allowed us to obtain 242 more projects than with the initially defined window, thus starting the filtering stage of the study with 282 MS projects as shown in Fig. 8.

4.4 Subjects

The subjects of the study were open-source software projects. The *source of data* for the study is the data set published by Amoroso d'Aragona et al. (2024). The source population of their collected list of microservice projects was extracted from the World of Code (WoC). The WoC is a computational and statistical Open Source Software infrastructure that provides a research-ready, operational, updatable, and expandable dataset (Ma et al. 2021). We

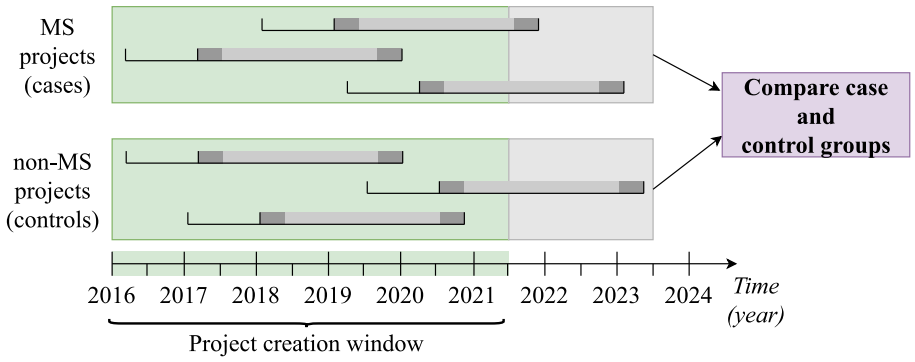

Fig. 2 The design of the retrospective cohort study conducted in this paper

considered adopting the cited dataset since it considered WoC as their source population data, and further it already considered part of the inclusion criteria we reported in the registered report (Saarimäki et al. 2023) such as *the minimum number of three MS* to consider an MS architecture, for example. However, all the projects included in the dataset are based on Docker and the dataset also lists projects which do not use the MS architecture. It should be noted that WoC is a high-dimensional dataset curated by thoroughly collecting and cross-referencing project objects (e.g., authors, projects, commits, blobs) mainly from three public version control systems, for instance, GitHub[5] which we consider in our study. We selected the cited data set because it already filters the large amount of projects in WoC using the following inclusion (I) criteria:

- I_1: Systems with at least one commit during 2021-2022, i.e. the last two years before the MS dataset creation.
- I_2: Systems with at least 100 commits.
- I_3: Systems with a community size of at least three contributors.
- I_4: Systems with a minimum number of three docker-based MS.
- I_5: Systems with more than 12 active months.
- I_6: Systems at least one year old.
- I_7: Systems with an existing README file written in English.

To provide high-quality data that would confirm if projects would be developed with an MS architecture, the filtering process included additional exclusion (E) criteria and excluded systems:

- E_1: Systems without information in the README file.
- E_2: Systems based on collections of tools/software/books.
- E_3: Systems *archived*.
- E_4: Systems that do not demonstrate an MS architecture.

The suitability of the identified projects was ensured by applying *eligibility criteria* that included both inclusion and exclusion criteria. The **inclusion criteria** determine the requirements each subject must meet to be included in the study sample. The following two criteria were used:

- An open-source project whose first commit was created on GitHub during 1/2016-6/2021.
- The project utilizes GitHub as its issue tracker.

Exclusion criteria define subjects who are excluded from the study despite meeting the inclusion criteria. The following criteria were applied:

- *Project had only one or two MS*. We defined a consolidated microservice project as a project that has at least three MS. Microservice projects with fewer than three services were not considered as cases or controls, as they do not fully adopt the MS architecture nor are large enough to be considered in this study. The number of MSs was identified from the *docker-compose* file, and files such as Dockerized databases and volumes were not included in the count.
- *The main development language of the project is not considered a proper programming language.* The main language of the project should be a programming language, that can be used in professional software development projects. To achieve this, we used the TIOBE Index [6] as a criterion to consider a language as a "programming language" criteria

[5] https://github.com

[6] https://www.tiobe.com/tiobe-index/programminglanguages_definition/

and excluded projects whose main language was not in the TIOBE index. The TIOBE index requires three criteria from the included languages. First, a language must have an entry in Wikipedia that clearly states that it is a programming language. Secondly, it is *Turing complete*, and thirdly, it should have at least 5,000 hits for Google search +"<language> programming". The index contains 358 programming languages, which can be examined from the TIOBE documentation and a list provided in the replication package of this study.

In addition to the selection criteria, we define a **loss to the follow-up criteria**. The criteria define the subjects who are excluded from the study based on their (in)activity during the follow-up period. Subjects who did not meet the criteria at the end of the follow-up period were considered dropouts and excluded from the final study subjects.

- *Use of MS or Docker was consistent during the follow-up period.* The criteria ensures the case projects were consistently using the MS architecture. To ensure their MS architecture during the follow-up period, we cloned the MS projects and performed identical analysis as conducted in the MS dataset (Amoroso d'Aragona et al. 2024) in all commits within the default branch registered during that time. We decided to exclude projects presenting less than the required number of MS in more than half of the analyzed commits, since it would demonstrate poor consistency in their MS architecture. This criterion ensured that MS projects were constantly exposed throughout the follow-up period. Moreover, interrupted usage of Docker ensures that no changes were made to the deployment technologies. Projects that changed their characterization technologies might introduce some bias in the evaluation due to the overhead of the adoption of the new technology itself.
- *The overall trend of commits decreased during follow-up*. The requirement ensures that the project is actively developed during the follow-up period. When a project is not developed, the code base does not change during follow-up, and it is not possible to detect any differences in the dependent variable. The trend and its direction were defined using the Mann-Kendall nonparametric trend test and Linear Regression from the development activity data collected during the follow-up period, i.e., the second year of activity since their creation. Projects showing an overall negative trend during this time were excluded. The method used to define the trend is explained in more detail in the Appendix A.1.
- *The overall trend of issues decreased during follow-up*. The criteria ensured that tracking issues were a well-integrated part of the development process. This was important as the dependent variable was calculated based on the reported issues. We considered the same approach used for the commit activity trend to measure the issue trend.

4.5 Variables

The Section describes the variables and hypothesized relationships included in the study. Figure 5 visualizes the relationships between the variables included in the study.

The **independent variable** defining the exposure is the *adoption of microservice architecture* as a boolean variable (indicating if a project uses MS or not). The study focuses on projects with MS architecture, deployed on Docker containers. A project was considered to have adopted the MS architecture if the repository contained at least 3 MS at the age of 12 months (at the start of the follow-up period). Figure 3 provides a detailed graphical representation of the follow-period at the project level. The number of MS was calculated using the same criteria as Amoroso d'Aragona et al. (2024). The adopted criteria are based on the

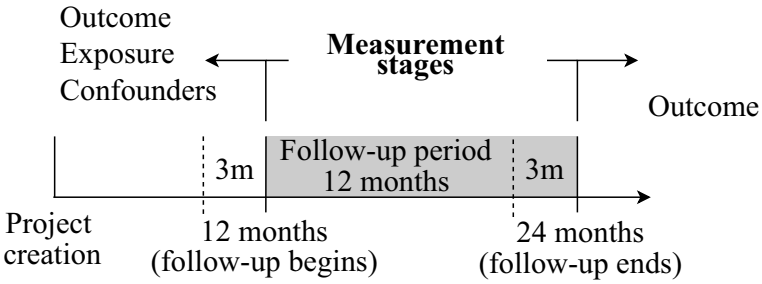

Fig. 3 Graphical representation of the follow-up period and velocity calculation windows at the project level

analysis of *docker-compose* and *Dockerfile* files created in software projects using Docker. From the mentioned files, the criterion analyzes the interconnection among services within the Docker framework, and based on the result *service dependency graph* the number of MS identified is calculated. The *docker-compose* configuration file provides documentation on the building process of a project to organize multiple images for containers. Furthermore, it contains relevant information about the services, containers, and networks displayed within the project.

The rationale for selecting projects with at least 3 MS was that applications with only two MS do not face the same issues as larger microservice applications. For example, coordinating three teams requires more extensive coordination than two, especially in decision-making. The same effect is also true when considering co-changes in MS (Soldani et al. 2018; Newman 2015). As recommended by Newman, if an MS system only needs two MS, the system may not have been properly architected, and a non-MS system might be a better alternative (Newman 2015).

The **dependent variable**, and therefore the outcome, was defined as the *velocity of a project* at the end of the follow-up period. The velocity at a specific point in time is determined by analyzing the issues closed over the preceding three months. Specifically, it represents the average duration between the creation and closure dates of those issues, measured in days, and therefore a lower velocity indicates a faster issue-fixing time. Figure 4 presents an example of a velocity calculation. A 3-month window was selected as the study followed projects from their 12th month after their creation. A shorter window was considered to have potentially too few issues, while six months would have been too large a proportion of the overall life of young projects.

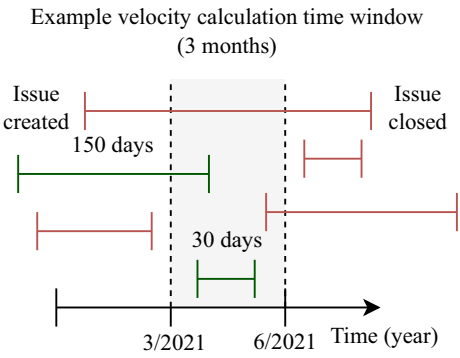

Fig. 4 An example of velocity calculation for time point 6/2021. The green issues are used in the calculation as they are closed between 3-6/2021, while the red ones are excluded because they are closed before or after. The velocity is the average time the included issues were open: (150 + 30) days / 2 = 90 days

We considered nine **confounding variables** in the study (Fig. 5). However, we acknowledge that selecting the most appropriate set of confounders is a complex task commonly faced in observational studies and it is not possible to include all confounders. Therefore, we discuss this issue in depth in Threats to Validity (Section 7). All included confounders are measured at the start of the follow-up period. The following variables were included in the study:

- *Start velocity*. The project velocity at the start of the follow-up period. Required for obtaining temporal precedence as the dependent variable is continuous. The velocity at the start of the follow-up is assumed to have a relationship with the end-velocity of the project, e.g., projects fixing issues quickly when they are one year old probably fix them quickly also at the end of the follow-up.
- *Size of the project*: The size of the project was determined by calculating the number of lines of code in languages included in the TIOBE index, based on a single commit. To obtain the total number of lines of code, the Git repository of a project was mined and the lines of code of the programming languages present in the repository were counted. Similarly, languages that were not included in the TIOBE index were filtered out, and therefore their respective lines of code were not included in the final number.
- *Development language*: The main development language of a project was defined as the programming language included in the TIOBE index having the highest number of files in the commit closest to the confounding variable collection date. The number of files was used as the ranking variable instead of lines to mitigate the differences between languages. They have varying purposes and syntax structures, so the number of lines produced in a language can vary significantly and, therefore, the number of lines of code in a project could be greater for a supporting language than for the main development language because of syntax differences.
- *Number of programming languages*: A higher number of different programming languages might fragment the development community, and therefore reduce velocity in case no developers are able anymore to modify a service written in a specific language. Languages non-identified within the TIOBE index are excluded from the final number.
- *Number of commits*: Total number of commits in the GitHub repository of a project. Each commit might create or fix an issue, and therefore, it could affect the velocity. Since some of the issues could be reported in different branches, and we also want to capture the overall development registered until the start of the follow-up period, we considered the commits from all the branches in GitHub.
- *Number of issues*: Total number of issues created in the GitHub repository of the project. We consider open and closed issues since we want to capture the overall issue development activity.
- *Number of developers*: The number of GitHub usernames that have made commits to the GitHub repository. Automated contributors such as *dependa[bot]* were excluded. In theory, more developers should be able to develop more code. However, the higher the number of developers, the longer the count of communication needed, thus slowing down the development process (i.e., lower velocity).
- *Creation year*: The year in which the projects registered their first commit in GitHub. The older the project, the higher the legacy that is commonly inherited. Younger projects might adopt more modern development practices. On the other hand, we expect that developers working on older projects might be accustomed to specific methods, tools, or business processes.

In addition to the collected confounders, in the design phase of the study, we also considered the variable *Age*, but it was excluded from the study design. The study followed the development of software projects during the first two years after their creation. Therefore, all projects were of the same age at the start of the follow-up and the variable age would not contain information. Similarly, we hypothesized that the interaction between the number of developers and the size of the project could affect the dependent variable. This relationship is not included in the study as an additional variable, but as an interaction effect.

– *Size / #Developers*: A proxy for the amount of code that a single developer produces. Projects developed largely by a very limited number of developers might have a different velocity than projects developed by many developers.

4.6 Data Sources and Measurement

We then extracted all the data required for the list of projects obtained from the dataset. We started the data filtering process by crawling the commit and issue history of the projects from GitHub as we already knew it to be the default issue tracker from the criteria followed in the adopted data set (Amoroso d'Aragona et al. 2024). We used the GitHub REST API to crawl for commits and issues from all branches, since we were interested in collecting the entire commit and issue activity of the remaining projects. The data were collected by crawling from the initial commit until the final commit made before the project reached years of age.

As the GitHub API does not provide the size of the projects, and also, we needed to collect the main programming language and the number of programming languages used within the first year of the life of the projects, we considered using the Sloc Cloc and Code (SCC) tool (Boyter 2018) to collect these metrics. To accomplish that, we cloned the remaining projects from both groups and analyzed the status of each project at the commit from the default branch, which was registered the closest before the beginning of the follow-up period.

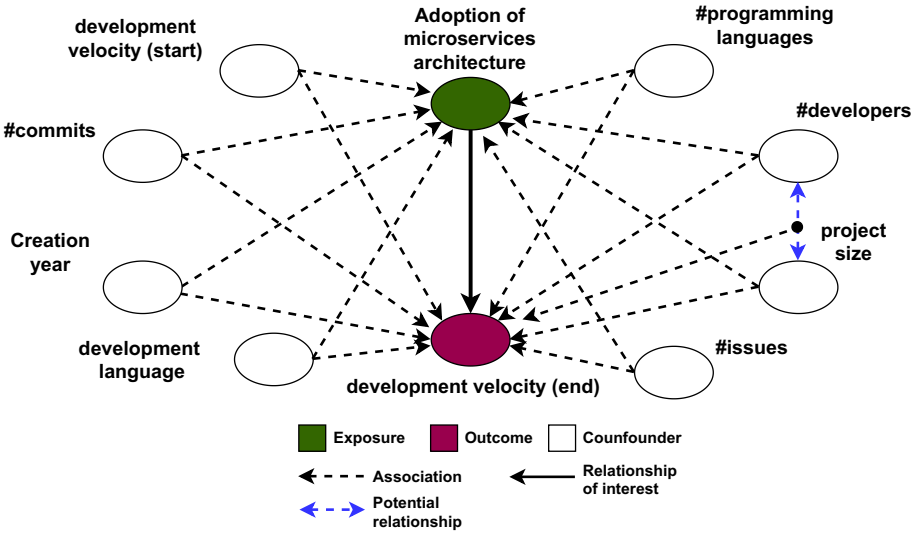

Fig. 5 Graph visualizing the included variables and their (hypothesized) connections

We considered the SCC tool as it provided a large language support among other reasons that prove its consistency. However, SCC considered languages that do not exist in the TIOBE index such as HTML or XML, therefore, we did not consider the excluded languages or their code lines in the calculation of the *size* and *number of languages* variables for the study. Furthermore, the SCC ranked the programming languages based on the number of files detected in the repository during the analysis instead of using the number of lines of code given the syntax difference between the programming languages.

4.7 Study Size

The data for this study originate from GitHub, which has a very large number of projects. Although starting with these large sources of data, the eligibility criteria used in the adopted data set (Amoroso d'Aragona et al. 2024) and the additional criteria posed by this study could result in a relatively small set of projects. Detecting a difference between two groups requires a sufficient number of subjects.

Power analysis is a statistical estimate performed to determine the minimum sample sizes required to detect significant effects from the available data (Ellis 2010). It ensures that the study results are valid with an estimated level of confidence and can detect meaningful effects. Power analysis requires defining the power, significance level, and estimated effect size for the study. Power is the probability of rejecting the null hypothesis when it is false, while significance level determines the probability of rejecting the null hypothesis when it is true. Initially, before the study had been executed, and therefore before we had observed the considered source of data, we used the value 80% for the power and 5% for the significance level, both commonly adopted values. The size of the relevant effect in this study was determined as small (Cohen's $d \geq 0.2$ (Cohen 2013)). Although MS is adopted mainly for reasons other than improving velocity, it is expected to affect it. Having up to four controls for each case can increase the statistical power of a study (Song and Chung 2010). Therefore, we calculate the sample sizes required for several case-to-control ratios. The calculation of the sample sizes based on the case-to-control ratios was carried out using power analysis for the "Independent-samples T Test" functionality since we initially assumed that the sample data were normally distributed. The results of the t-test for the case-to-control ratios we expected are presented in Table 2.

As the study design has strict selection criteria for the study subjects, the sample obtained may be smaller than required based on the power analysis. To understand and quantify the impact of loss of subjects, we performed a power analysis and effect size calculation on the resulting sample size. The original sample size calculation assumes that the data are normally distributed; however, real-life data is often non-normal. Therefore, we adopted a generalized method to conduct the power analysis for the t-test using Monte Carlo (MC)

Table 2 Results from the power calculation for the expected different case-to-control ratios

	Case to control ratio			
	1:1	1:2	1:3	1:4
#Cases	394	295	263	246
#Controls	394	590	787	983
Total	788	885	1,050	1,229

Power=0.8, p=0.05, Effect size=0.2

simulation (Du et al. 2017; Zhang 2014). The adopted approach estimates the power through the empirical distribution found on the first four moments of a distribution, i.e. (*mean* (μ), *standard deviation* (σ), *skewness* ($\gamma_1 = E[(\frac{x-\mu}{\sigma})^3]$), and *kurtosis* ($\gamma_2 = E[(\frac{x-\mu}{\sigma})^4]$)), as well as the sample size of each group. From the simulations derived, the statistic t^* is calculated for each simulated set. Thus, the power is estimated as the proportion of t^* statistic values that were greater than the critical value ($\alpha = 0.05$). A further description of the simulation-based technique adopted can be found in the Appendix A.3.

Similarly, we estimate the effect size for the study sample obtained. Unfortunately, this also suffers from the same issue as power analysis, and there are no robust nonparametric techniques to estimate the effect size. Therefore, if necessary, the data were transformed to fit the normal distribution and compute Cohen's effect size statistic d (Cohen 2013). Cohen evaluated the quantification of the effect size magnitude using three thresholds, i.e. $|d| < 0.2$ "*negligible*", $|d| < 0.5$ "*small*", $|d| < 0.8$ "*medium*", and otherwise "*large*". However, based on the results obtained in Section 5.1, we did not assume any minimum effect size and relied on the robustness of the resulting sample groups.

4.8 Statistical Methods

The section describes the methods adopted for conducting the statistical analysis of this study. Data analysis is an iterative process in which methods are selected based on the data collected. This results in the methods and results being tangled. One of the main characteristics of the data that guided the analysis was that the data were skewed rather than distributed normally (see Fig. 12 and Table 4). Similarly, the unexpected decrease in the number of study subjects remained a major threat to the effect of the results thrown from the analysis, as will be described throughout Section 5 and will be further discussed in the threats to validity (see Section 7).

Understanding if a real relationship exists between the independent and dependent variables requires also investigating their connections with confounding variables. Therefore, the analysis covers all three connections shown in Fig. 6. The analysis was performed in two different stages, first, the *crude analysis* considered only the unadjusted raw relationship between the cases and controls. In the second step, the confounders were included in the analysis as in cohort studies, the confounders need to be treated to obtain a non-confounded relationship between the independent and dependent variables. To perform such a statistical adjustment to capture the effect of the selected confounding variables, these, along with the independent variable, are considered as predictors when modeling the relationship to explain the behavior of the dependent variable.

The goal of the analysis is to understand whether the use of MS architecture (an independent variable explaining the exposure) has an impact on the velocity of young software projects. We considered the velocity for closing issues during the second velocity calculation window as the dependent variable and, therefore, we considered the analysis of mean differences as the statistical approach to measuring the difference between projects using MS

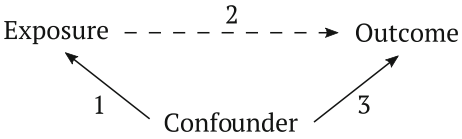

Fig. 6 The relationships between independent variable, dependent variable, and confounder

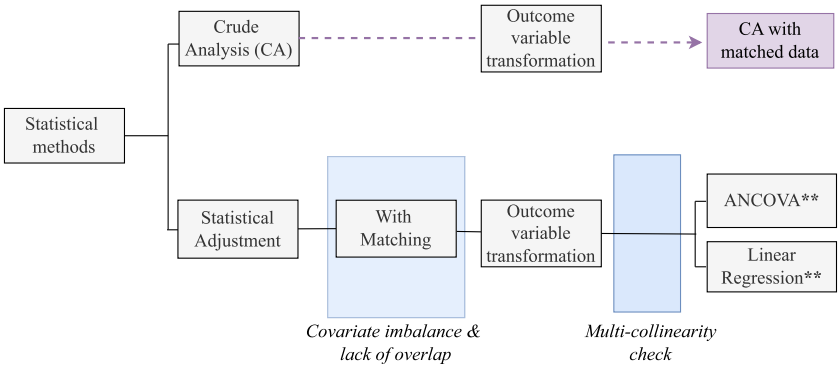

Fig. 7 Design of the statistical analysis performed in this study. (**: Analysis of the normality of the residuals required for model validity assessment.)

architecture and those using non-MS architecture. Figure 7 offers an illustrative overview of the statistical analysis design used in this study.

4.8.1 Descriptive Analysis

Detailed analysis is conducted mainly using common methods such as boxplots, histograms, QQ-plots, and density function plots. However, as in other scientific fields, such as social and behavioral sciences, and also in software engineering, the practical data are rarely normally distributed. Therefore, the normality of a distribution of the dependent variable was tested using nonparametric tests of the Shapiro-Wilk (SW) (Royston 1982) and Anderson-Darling (AD) (D'Agostino 2017) nonparametric tests. Both test statistics are well known for defining the hypotheses in the following format:

- H_0: There is statistically significant evidence on the normality of the data distribution.
- H_1: There is no statistically significant evidence to determine normality in the data distribution.

The SW test is known to be the test with the highest power among the existing probability tests (Razali et al. 2011). However, there are several discussions about the suitable sample size limit required to run it. Therefore, we also calculated the AD test, which provides slightly smaller power, but is not constrained by the sample size of the test (Razali et al. 2011).

4.8.2 Crude Analysis

The first step of the analysis was to determine the unadjusted relationship between the cases and controls (arrow 2 in Fig. 6). The analysis did not include any confounding variables and the data were not transformed. The results serve as a foundation for further analysis, including the confounding variables. Given the lack of normality in the distribution of the end velocity in each of the study groups, the comparison was performed using the nonparametric unpaired independent samples *Wilcoxon rank sum*, more commonly referred to as the *Mann-Whitney* (MW) test (Hollander et al. 2013). The MW test compares the medians of the groups considered, and acts as the nonparametric alternative to the parametric *Student's t-test*, which we would have used if the data had been normally distributed from the beginning. In the context of our RQ, we defined our *two sided* hypotheses as follows.

- H_{0MW}: No difference between the median of the end velocity means for the cases group and the median for the end velocity means of the controls group.
- H_{1MW}: Difference between the median of the end velocity means for the cases group and the median for the end velocity means of the controls group.

The MW test is commonly used in clinical trials to assess the efficacy of two treatments or the effect of a certain exposure when the data are not normally distributed (Hart 2001). However, the use of the MW test for non-normal data is misleading as, although there are many situations where the MW test has more power and is more efficient than the parametric counterpart *t-test* (Blair and Higgins 1980), it assumes that the data are independent random samples from two populations that have the same distribution shape. The distribution shape of the dependent variable depicted in the box plot format of Fig. 9 strongly supported this view. However, the spread of the observations along the distribution of the non-MS projects group in both end and start velocity mean observations suggested that the difference between the groups would exist only on the spread of the distributions (Hart 2001).

Consequently, we also considered performing the crude analysis with *transformed* data. Transforming the distribution of the dependent variable into a normal distribution would allow us to perform parametric tests (Faresjö and Faresjö 2010). This stage of the analysis will be further discussed later in this section. With a normally distributed dependent variable, we consider running the *independent sample t-test*, which compares the means of groups. Thus, we defined the test hypotheses through the definition of hypothesis *two-sided* (Moore and McCabe 1989):

- H_{0T}: There is no difference between the mean velocity of the cases group and the mean velocity of the controls group
- H_{1T}: There is difference between the mean velocity of the cases group and the mean velocity of the controls group

Nonetheless, the presented hypothesis definition allows us to understand whether the mean of the case group is different than that of the control group, that is, whether the velocity in closing issues is different or not.

4.8.3 Adjusted Analysis

The second stage of the analysis investigates the adjusted relationship between the case and control groups. This requires treating the included confounders to remove their effect from the observed relationship between the independent and dependent variables. Therefore, confounders and independent variables are included in the modeling stage as predictors to estimate their impact on the behavior of the dependent variable.

In causal inference, there are certain situations in which it is dangerous to use a standard linear regression as the outcome of the predictors and an indicator variable to estimate causal effects, for example, the confounder *imbalance* and *lack of complete overlap* (Rosenbaum 2002). The confounder imbalance occurs if the distribution of relevant confounders differs for the case and control groups. Likewise, a lack of complete overlap occurs if there are regions in the space of relevant confounders where there are cases but not controls or controls but no cases. Given the shape of the distribution presented by the selected confounders, in Table 3 that only *Main language* had a lack of complete overlap.

Matching *Matching* ensures the study population contains n similar controls for each case and is done in the design phase of the study. For example, for each project in the case group, three projects of the same domain and size are included in the control group. This ensures that

the case and control groups are similarly exposed to the effect of confounders. The matching criteria can consist of one or a combination of several variables and the matching can be conducted using several different algorithms (Rosenbaum 2010). However, matching prevents studying the variables used in the process (Gordis 2014). To avoid confounder imbalance and lack of complete overlap, and considering the results of the descriptive analysis, we performed a *matching* analysis on the variable *Main language*.

The matching stage involves three main steps: planning (I), matching (II), and evaluating the quality of matching (III). It is important to note that the process is iterative and is repeated until the quality of matching is good enough or the matching is determined as an unsuitable treatment method for the variable. The planning stage consists of selecting the effect to be estimated (which in the case of this study is the independent variable) and performing an initial descriptive analysis on the selected confounders to identify which of them denote confounder imbalance and/or lack of overlap (see Section 5.2). Then, the matching technique is selected based on the selected variables. We used *exact matching* as only one confounder was required to be balanced, and this technique completely removes the confounding effect of the included variable (Stuart et al. 2011). With exact matching, a complete cross of the included confounders is used to form subclasses defined by each combination of the confounder levels. Any subclass that does not contain both cases and controls is discarded, leaving only subclasses containing cases and controls that have the same level for the matched confounding variables. In step II, the data are matched according to the planning phase. Finally, Step III assesses the quality of matches by analyzing the impact of the performed match on the confounder balance. As balance metrics we used *Standardized Mean Differences* (SMD), *Variance Ratio*, *Empirical CDF statistics*, and *Visual Diagnostics*. We expand the mathematical description of the adopted metrics in the Appendix A.4.

Matching produces the subjects matched from the case and control groups, as well as the *matching weights* for each subject. The weights for the exact matching are based on the *subclass propensity score* (SSP). The SSP is calculated as the proportion of units in each subclass that are in the case group, and all units in that subclass are assigned to Stuart et al. (2011) (do not relate to *the matching of the propensity score*). Thus, the weights are calculated based on the *inverse probability weight* standard formula, which assigns 1 for the cases, $SSP/(1 - SSP)$ for the control units, and 0 for the unmatched units. The computed weights are then included along with the data of the confounders in the statistical adjustment stage to estimate the exposure effect and similarly quantify the confounder balance achieved through the matching (Stuart et al. 2011).

Estimating the Exposure Effect With the resulting distributions from the matching stage, we used statistical regression models to determine the effect of exposure on the dependent variable while adjusting the impact of the remaining effect of confounders. Confounders, excluding the matched confounder, were added to the models considered as additional independent predictors.

As stated previously, the observed distribution of the dependent variable was non-normal and, therefore, required transformation. The data showed exaggerated right skewness, so we selected the *cube root transformation* (Box and Cox 1964) $\left\{x \Rightarrow x^{(1/3)}\right\}$ which provides a fairly strong transformation with a substantial effect on the shape of the distribution (see Section 5). Furthermore, we did a *multicollinearity assessment* of the model predictors considered, that is, we remove the variables that present collinearity among each other in a multiple regression model. We exploited the *Variance Inflation Factor* (VIF) (O'brien 2007), which measures the increase in model variance for each variable and excluded variables that scored higher than 5 (Sheather 2009).

To determine the impact of the predictor on the results of the dependent variable, we considered parametric regression analysis methods based on the distributional assumptions observed from the transformed dependent variable, which will be explained later in Section 5. Concretely, we considered commonly used methods in the field of cohort studies such as *Analysis of Covariance* (ANCOVA), as well as the *Multiple Linear Regression* (Rosenbaum 2002). We describe the assumptions required for each model and the model assessment diagnostics later in Section 5.

4.9 Verifiability and Replicability

To allow verifiability and replicability, we made the raw data and analysis files available in our online appendix.[7]

5 Results

In this Section, we report the results obtained to answer our RQ. The results are presented following the STROBE guidelines (Von Elm et al. 2007) described in Table 1.

5.1 Study Subjects

The section describes the results of the subject filtering process. The general process defined in Section 4.4 and the number of subjects included and excluded at each step are visualized in Fig. 8.

The MS data set contained 378 MS projects (cases) and 53,842 non-MS projects (controls), and we excluded subjects from that by applying the exclusion and follow-up criteria. Step I of the subject selection process excluded projects that were created outside of 1/2016-6/2021, resulting in 282 MS projects and only 1,052 non-MS projects which were created during this time window. This first exclusion step reduced the sample sizes by 96 (24.8%) case projects and 52,790 (98%) control projects. The drastic reduction, especially in the number of non-MS projects, was caused by two main factors. First, some of the projects were indeed created outside of the defined window, and second, some projects did not register any issue during their first two years of existence in GitHub, therefore lacking a consistent development in the early years of activity. Considering that using GitHub as the default issue tracker was the prerequisite for their inclusion in the source data set (Amoroso d'Aragona et al. 2024), we assume that the lack of registered issues in excluded projects is due to non-existing activity rather than the use of external issue tracking systems.

The subject exclusion step II removed projects that had negative issue tracking or commit development activity. Testing different aggregation windows for the commit and issue tracking data proved the quarterly aggregated data to be the best-suited as it provided the highest number of projects fulfilling the defined criteria. We obtained 101 MS and 318 non-MS projects having a non-negative activity trend in commit and issue generation.

Additionally, we ensured the usage of Docker and MS during the follow-up period of each MS project, i.e. their second year of life. All remaining 101 MS projects demonstrated sufficient MS architecture, and therefore the check did not exclude any projects.

[7] https://doi.org/10.5281/zenodo.14876471

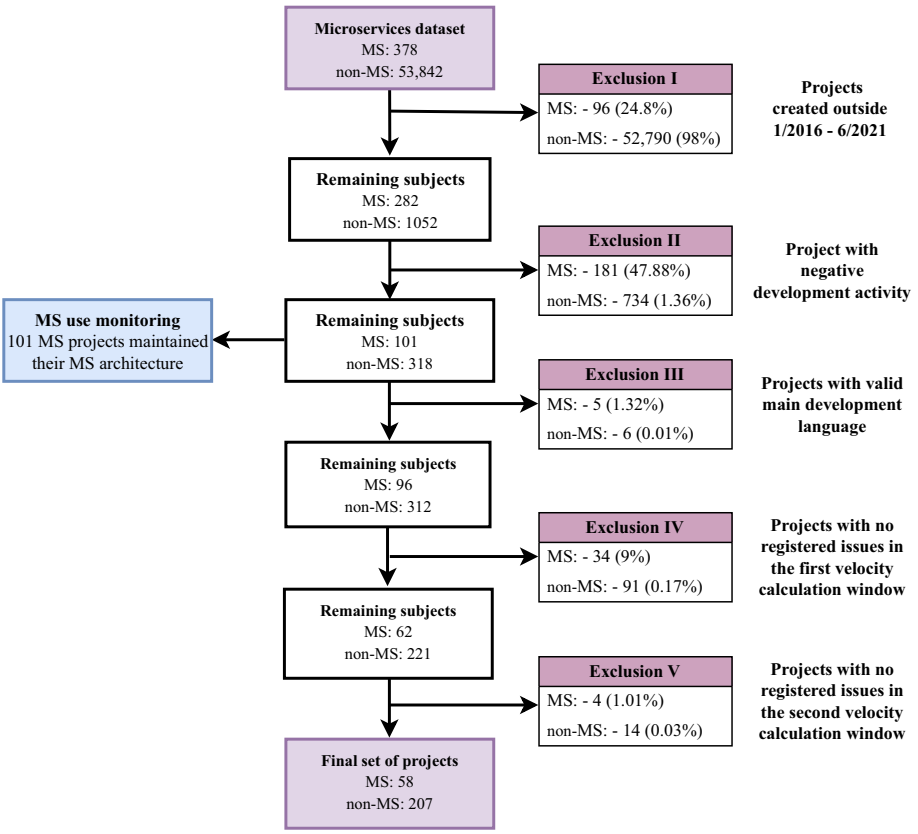

Fig. 8 The design of the filtering stage of the study

Exclusion Step III excluded 5 MS projects and 6 non-MS projects that did not contain any languages included in the TIOBE index.

To calculate the velocity at the start and end of the follow-up period, we considered the lifetime of the registered issues fixed during the velocity calculation periods depicted in Fig. 2. Subject exclusion steps IV and V excluded 34 MS projects and 91 non-MS projects during the first velocity calculation window, and 4 MS projects and 14 non-MS projects during the second window, as they did not register any issue closures within the defined periods. The high number of exclusions in the first velocity calculation window for both groups indicates a lack of registered issues in the early stages of development compared to the second window, where projects are almost two years old. The excluded projects showed minimal issue-related activity after their initial commit on GitHub, resulting in fewer opportunities to close issues within the specified calculation windows. However, the shorter time frame for admitting eligible issues during the velocity calculation stage may have contributed to the high rejection rate, posing a threat to validity due to the limited number of analyzable projects.

The final study sample consisted of 58 MS projects and 207 non-MS projects. Consequently, the subject selection process reduced the populations of the case and control groups by 85% and 99.6%, respectively, compared to the initial numbers collected from the MS dataset. We provide basic statistics of the resulting projects in the online appendix reported

in Section 4.9. This includes information such as project size, the number of MS components, stars, and the number of issues and commits among others at the start and end of the follow-up periods.

Power analysis and effect size calculation Based on the resulting projects for each of the study groups, the calculated power analysis resulted in a statistical power of 0.055 (Cohen 2013). Similarly, the effect size computation yielded a small effect size of 0.17. These results are far from the expected power and effect size assumed in Section 4.7. However, it should be noted that despite the substantial reduction in data points, the resulting data provide a high-quality subset of projects. The two groups of projects initially collected from the MS dataset were the result of exhaustive data mining, resulting in a manually curated list of consistent MS projects from the entire existing data on GitHub. The multiple exclusion criteria applied in this study ensured that only active and suitable software projects at the early stages of their evolution were considered for the analysis. Therefore, we emphasize the importance of the trade-off between power and data quality, given the resulting sample size. This ensures the high quality of the observational methodology carried out in the study and provides a more exhaustive subset of MS projects capable of demonstrating robust development activity in their early stages. The data and scripts used to extract them are publicly available in the replication package 7.

5.2 Descriptive Data

Before estimating total causal effects, we first measured all confounder variables based on case groups to check potential confounder imbalance or lack of overlap (VanderWeele 2019).

Variable distribution Figure 9 presents the distribution shape and spread of the dependent variable for the two study groups. Similarly, Fig. 10 shows the same projection for the continuous confounders considered for the case (MS) and control (non-MS) groups, while Fig. 11 and Table 3 provide descriptive insights from the categorical confounding variables *Creation year* and *Main programming language* respectively.

The distribution of the dependent variable in both the case and control groups shows a clear skewness (Fig. 12). Table 4 demonstrates that there is strong evidence to reject H_0 based on both of the considered normality-tests.

The descriptive analysis showed a clear lack of normality in the model predictors. We observed a reasonable confounder balance among most of the confounding variables. However, we found a lack of overlap between the study groups in the main *programming language* with programming languages such as Elixir, C / C++, and F #, among others. Table 12 shows

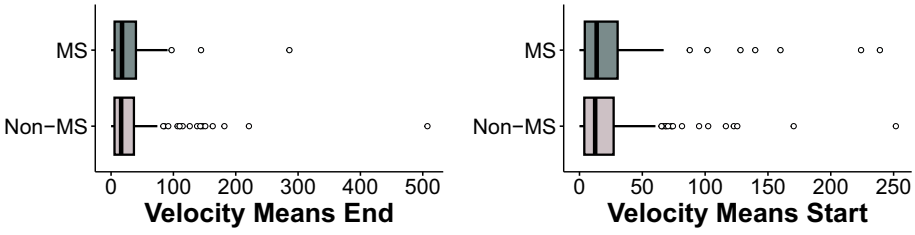

Fig. 9 Spread and distribution shape representation of the velocity mean calculations at the first and second (dependent variable) calculation windows

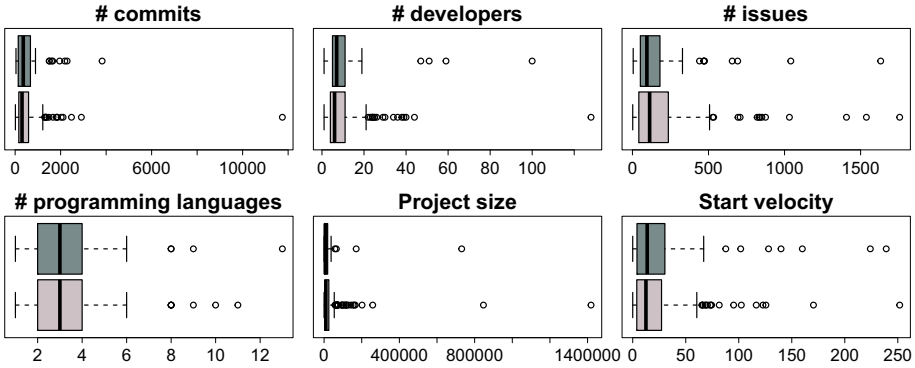

Fig. 10 Exploratory visualization of the continuous confounder variables. (#: *Number of*, cyan: MS projects, lavender: Non-MS projects)

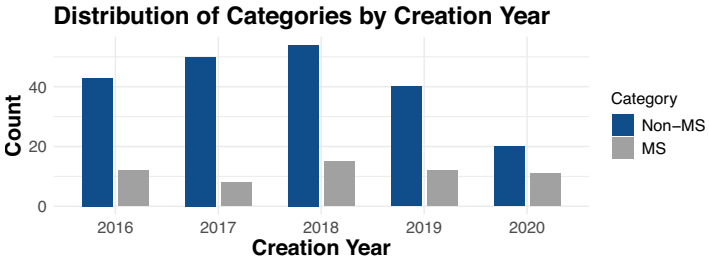

Fig. 11 Exploratory visualization of the categorical confounder Creation Year

Table 3 MS and Non-MS projects distributed by main programming languages

Language	MS	Non-MS	Language	MS	Non-MS
Python	13	55	JavaScript	11	36
Java	8	14	PHP	6	12
C#	6	8	Go	4	26
TypeScript	3	20	Rust	2	2
Scala	1	2	BASH	1	1
Shell	1	6	Ruby	1	16
Elixir	1	0	C	0	3
C++	0	3	SQL	0	1
ClojureScript	0	1	F#	0	1

(Grey : Lack of overlap)

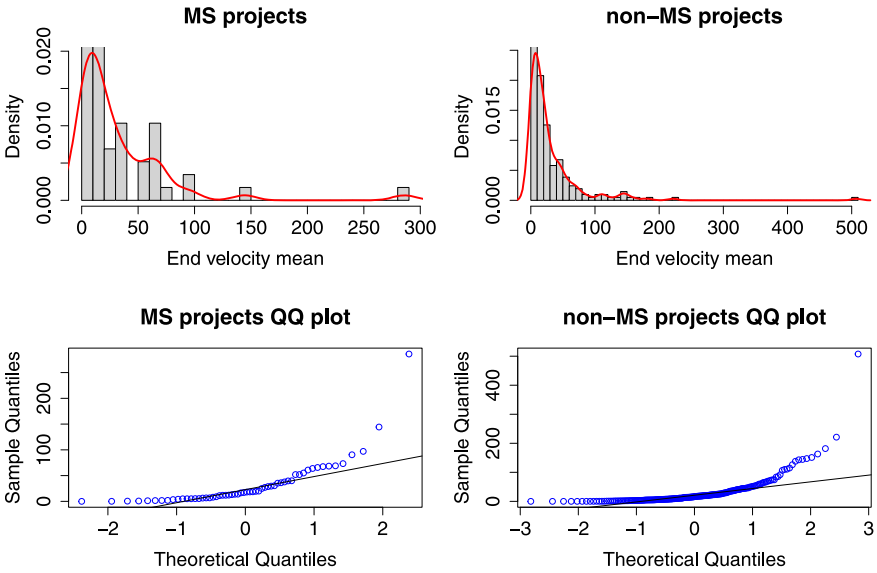

Fig. 12 Probability density function plot and QQ plot of the dependent variable of each study group

the initial values of the confounder imbalance metrics considered. In general, we observed reasonable mean differences between the case and control groups, as well as most of the SMD values located close to zero with some exceptions such as the *Creation year* on the categories 2017 and 2020, or *Main language* on languages such as Ruby or Go. Similarly, both the mean eCDF and maximum eCDF differences showed small values, adding more robustness to the acceptable confounder balance observed in the SMD values. However, the variance ratio reported by confounding variables such as *Velocity Start* with 2.87, *# Programming languages* with 1.84, and *# Developers* with 2.02 denoted a clear deviation from the unity value 1.

Matching Considering the observed descriptive data and the resulting number of subjects reported, we decided to initially implement *exact matching* on the confounding variable *main programming language*. Table 5 presents the matching results after the exact matching analysis with a final output of ten unmatched subjects and therefore discarded. Thus, 198 control subjects (non-MS) and 57 case subjects (MS) were used to estimate the effects of independent variables.

After performing the matching, we assess its quality. The main results are presented in Table 6 which presents the impact of the match on the selected confounder balance metrics. Despite the loss of 10 subjects from the study groups, the impact on confounder balance

Table 4 P-values from the performed Shapiro-Wilk (SW) and Anderson-Darling (AD) normality tests

	SW (p-value)	AD (p-value)
End velocity (MS, cases)	$1.079e^{-10}$	$1.508e^{-12}$
End velocity (Non-MS, controls)	$< 2.2e^{-16}$	$< 2.2e^{-16}$

Critical value (α)=0.05

Table 5 Results from the exact matching based on Programming Language

	Controls	Cases
All	207	58
Matched	198	57
Unmatched	9	1

appears negligible. We also performed additional analyzes to assess the quality of the matching, and they are presented in the Appendix B.

In addition to confounder balance, the quality of the match is determined by how many units remain after matching, since the effect estimate may also be imprecise. In this type of scenario, the trade-off between confounder imbalance and exposure estimation requires subjective judgment about the study case (Ho et al. 2007). Therefore, given our initial reduced population of subjects due to the exhaustive subject filtering performed, we decided to run the estimation of the effect of the independent variable without further matching to not lose more subjects. Moreover, guidelines indicate that 0.1 or 0.25 represent reasonable cut-offs for acceptable Absolute SMD while larger standardized biases would indicate that groups are too different from one another for a reliable comparison (Stuart et al. 2013; Rubin 2001).

To estimate the effect of the independent variable, that is, exposed, we first perform the data preprocessing stage previously described in Section 4.2. Table 7 provides the results of the normality test performed with the *Shapiro-Wilcox* and *Anderson-Darling* tests, following the same structure used in Section 4.7. The results showed statistical evidence of the normality distribution of the MS projects after performing the cube root transformation. However, such adopted transformation commonly used for right-skewed data was not able to provide a statistically significant probability to prove normality in the non-MS projects' distribution. However, given the considerable improvement in the resulting test probabilities considering the initial results in Table 4, we considered testing the existence of the Central Limit Theorem (CLT) in the context of the distribution *End Velocity* of the non-MS group. The CLT states that, under certain conditions, the distribution of a normalized version of the sample mean converges to a standard normal distribution (Casella and Berger 2024). Therefore, the theorem

Table 6 Summary of the balance of all matched data

	MCa	MCo	SMD	VRatio	eCDF Mean	eCDF Max
Velocity Start	32.35	21.75	0.21	2.87	0.04	0.10
Project Size	27,075.29	34,812.02	-0.08	0.67	0.08	0.14
# PL	3.40	3.10	0.14	1.84	0.02	0.06
CY= 2016	0.21	0.21	-0.00	.	0.00	0.00
CY= 2017	0.14	0.24	-0.30	.	0.10	0.10
CY= 2018	0.26	0.26	-0.01	.	0.00	0.00
CY= 2019	0.21	0.19	0.03	.	0.01	0.01
CY= 2020	0.19	0.10	0.24	.	0.09	0.09
# Commits	569.43	511.87	0.08	0.59	0.04	0.14
# Issues	186.67	193.27	-0.02	1.19	0.05	0.13
# Developers	11.81	9.42	0.15	2.02	0.03	0.12

PL = programming language, CY = creation year
(#: *Number of*, MCa: *Mean of the Cases*, MCo: *Mean of the Controls*, SMD: Standardized Mean Differences)

Table 7 Probability results from the performed Shapiro-Wilcox (SW) and Anderson-Darling (AD) normality tests with the cube root transformed data

	SW	AD
End velocity (MS, cases)	0.6096	0.7538
End velocity (Non-MS, controls)	0.001082	0.01932
CLT End velocity (Non-MS, controls)	0.6622	0.5459

Critical value (α)=0.05

relies on the assumption of an asymptotically normally distributed population if and only if the random sample shows normality.

Consequently, we implemented random sampling on the non-MS end velocity distribution for a sample size of 30 observations as recommended (Casella and Berger 2024). Table 7 demonstrated evidence of the application of CLTs in the sample distribution, therefore, assuming the asymptotically normal distribution of the population data.

Based on all the conducted analysis of the quality of matching, we concluded that the matching is of good quality and therefore conduct further analysis for the study using the matched study sample.

Multicollinearity Finally, we investigate the multicollinearity between the model predictors exploiting the VIF score. Based on the specified threshold, none of the model predictors scored more than 5 (*Number of Issues* reporting the higher score with 1.5); therefore, we did not exclude additional predictors before estimating the independent variable effect.

5.3 Main Results

The section presents the results from the analysis of the crude (unadjusted) and adjusted data.

5.3.1 Crude Analysis

The visualization of the dependent variable for the case and control groups in Fig. 9 depicted a lack of normal distribution. Therefore, and according to what is described in Section 4.8, we tested the unadjusted relationship between MS and non-MS architecture using the MW test. The defined *two-sided* MW test resulted in a *p-value* of 0.598, therefore providing not enough statistically significance to reject the null hypothesis H_{0MW}. We can observed the small diferences between the medians from both groups numerically in Table 8 and graphically in Fig. 9. Consequently, **we can claim that based on the crude analysis results there is no statistically significant difference between the velocity means of MS projects at the end of the observational period and that of the non-MS projects.**

As explained and justified in Section 4.8.2, we tested the unadjusted relationship through the parametric *dependent samples t-test* after performing the cube root transformation on the dependent variable. The test result reported a p-value of 0.678, therefore, there was no statistically significant evidence to reject the null hypothesis H_{0T}. Consequently, **we can**

Table 8 Median, Q1 and Q3 quantiles of the MS and Non-MS projects velocity means at the end of the follow-up period

Project type	Median	Q1	Q3
MS	18.10	6.38	40
Non-MS	15.30	5.03	35.70

Table 9 Mean difference quantification statistics for the crude analysis results

	Effect Size	CI (95%)
Mann-Whitney test	0.17	$(-\infty, 5.375)$
t-test	0.07	$(-\infty, 0.392)$

CI = Confidence Interval

claim there is no statistically significant difference between the mean velocity of the MS projects at the end of the observational period and the mean of non-MS projects. The reported results for the unadjusted relationship between the cases and control groups denote an overall balance between velovity observed in MS projects and non-MS projects.

Table 9 provides summary statistics such as effect size and the confidence interval of the mean difference to quantify the mean differences tested. The effect size values for the MW test were already calculated in Section 4.7 while the value for the t-test is calculated with the commonly used parametric method *Cohen's D*. Similarly, the confidence intervals are shown in *one-sided* format following the definition of the hypotheses we described in the design stage of this study. From the results obtained, we can observe that the estimated borderline values for the mean differences remain small, a result that supports the initial view of the dependent variable that we illustrate in Fig. 9.

5.3.2 Adjusted Analysis

The crude analysis indicated an existing unadjusted relationship between projects that have an MS architecture and a higher velocity (less time needed to resolve problems). To determine whether the crude result is affected by the collected confounders, we modeled the relationship using statistical models that included them. Following the design described in Section 4.8 and visualized in Fig. 7, we performed the statistical adjustment for confounders using parametric ANCOVA models and linear regression with transformed cube root data.

Table 10 presents the results for the ANCOVA model after performing the matching. From the provided table, we can highlight the *p-values* associated with the explainability of each of the model predictors, as well as the *F statistic* value to which the p-values are associated and that explains how well the predictor explains the variation of the dependent variable (Casella and Berger 2024). Furthermore, in the table, we present the degrees of freedom (*Df*), the

Table 10 Summary statistics for the ANCOVA model after matching

	Df	SS	MS	F value	p-value	η^2	P. η^2	P. ω^2	P. ϵ^2
MS/non-MS	1	1.67	1.67	1.09	0.2965	0.0043	0.0045	$3.70e^{-04}$	$3.87e^{-04}$
VS	1	2.94	2.94	1.93	0.1660	0.0076	0.0079	$3.63e^{-03}$	$3.80e^{-03}$
Size	1	2.74	2.74	1.80	0.1808	0.0071	0.0073	$3.13e^{-03}$	$3.27e^{-03}$
# Dev	1	2.33	2.33	1.53	0.2170	0.0061	0.0062	$2.08e^{-03}$	$2.18e^{-03}$
# PL	1	0.60	0.60	0.40	0.5300	0.0017	0.0016	0.00	0.00
CY	4	2.52	0.63	0.41	0.7982	0.0066	0.0068	0.00	0.00
# Commits	1	0.63	0.63	0.41	0.5203	0.0016	0.0017	0.00	0.00
# Issues	1	0.12	0.12	0.08	0.7779	0.0003	0.0003	0.00	0.00

PL = Programming Language, VS = Velocity Start
Dev = Developers, CY = Creation Year

sum of squares (*SS*) and the mean squares (*MS*) related to each predictor. In addition to reporting the statistical significance of the collected confounders, we also report the effect size of each predictor in order to quantify the proportion of the variance on the dependent variable explained by each confounding variable. We quantified this measure through the *Eta-Squared* (η^2) (Olejnik and Algina 2003), *Partial Eta-Squared* (P. η^2) (Olejnik and Algina 2003), *Partial Omega-Squared* (P. ω^2) (Carroll and Nordholm 1975) and *Partial Epsilon-Squared* (P. ϵ^2) (Allen 2017) statistics. We provide further description of the adopted effect size statistics in the Appendix C.

Given the removal of the categorical model predictor *Main Language* and the improvement in the confounder imbalance as previously explained, we would expect further explanatory results. However, the provided table does not highlight that the model predictors are considered statistically significant in explaining the behavior of the dependent variable. Moreover, the estimated effect size statistics reported small effect size values for all the confounding variables, that is, they explained only a small proportion of the variance. Therefore, based on observed values, **we cannot claim that the model predictor based on the use of the MS architecture is statistically significant to explain the variability of the dependent variable using ANCOVA after the matching has been performed**.

We observe the results of the linear regression model after matching in Table 11. The *Intercept* predictor generated by the model considers as baseline factor regression projects with the 2016 as *Creation Year*, and having a non-MS architecture, but does not consider any type of *Main Language* as this model predictor is removed after the matching stage. In the table, we include the results for the estimated value (*Estimate*) of each model predictor, the standard error (*Std. E*), as well as the resulting value for the t-statistic (*t value*). Similarly as with the results table from the ANCOVA model, for the linear regression model we also quantified the effect size of the model and the considered confounding variables. In order to assess the effect size of the model, we estimated the *Coefficient of Determination* (R^2) (Agresti 2015), which represents the proportion of the variance explained by the model predictors within the built model, as well as the *Adjusted* (R^2) (Agresti 2015), which penalizes

Table 11 Summary statistics for the Linear Regression model after matching

	Estimate	Std. E	t value	p-value	P. r^2	P. f	P. f^2
(Intercept)	2.2612	0.2340	9.66	$< 2e^{-16}$*	$2.77e^{-1}$	0.6198	$3.84e^{-1}$
Using MS	0.1459	0.1910	0.76	0.4457	$2.39e^{-3}$	0.0490	$2.40e^{-3}$
VS	0.0035	0.0022	1.60	0.1116	$1.04e^{-2}$	0.1024	$1.05e^{-2}$
Size	$8.82e^{-7}$	$9.05e^{-7}$	0.97	0.3305	$3.89e^{-3}$	0.0625	$3.91e^{-3}$
# Dev	0.0075	0.0083	0.90	0.3701	$3.31e^{-3}$	0.0576	$3.32e^{-3}$
# PL	0.0330	0.0529	0.62	0.5341	$1.59e^{-3}$	0.0399	$1.59e^{-3}$
CY - 2017	-0.0629	0.2329	-0.27	0.7873	$3.01e^{-4}$	0.0173	$3.01e^{-4}$
CY - 2018	-0.0134	0.2255	-0.06	0.9527	$1.45e^{-5}$	0.0038	$1.45e^{-5}$
CY - 2019	0.1354	0.2468	0.55	0.5838	$1.24e^{-3}$	0.0352	$1.24e^{-3}$
CY - 2020	-0.2433	0.2987	-0.81	0.4161	$2.72e^{-3}$	0.0523	$2.73e^{-3}$
# Commits	$-6.43e^{-5}$	$9.32e^{-5}$	-0.69	0.4911	$1.95e^{-3}$	0.0442	$1.96e^{-3}$
# Issues	$1.01e^{-4}$	$3.58e^{-4}$	0.28	0.7779	$3.279e^{-4}$	0.0181	$3.28e^{-4}$

PL = Programming Language, Dev = Developers, CY = Creation Year, VS = Velocity Start, * = Statistical significance

original statistics on the number of included model predictors and the sample size. To further quantify the impact of each confounder, we calculated the Partial r-squared (P. r^2) (Chin et al. 1998), the Partial f (P. f) (Cinelli and Hazlett 2020) and the Partial f-squared (P. f^2) (Cohen 2013) statistics. As reported previously, we provide further insights on the adopted effect size statistics in the Appendix C.

The reported p-values associated with the resulting t-value denote the statistical significance of the explanatory effect of each predictor in explaining the variation of the dependent variable. The resulting model reported R^2 and *Adjusted* R^2 effect size values of 0.0353 and -0.0083, suggesting that the model is capable of explaining only a small proportion of the variance of the dependent variable., therefore denoting a **small effect size** from the model. Likewise, the computed predictor-level effect size statistics contribute on supporting the conclusions made from the R^2 effect size results, that is, the effect size reported by each of the considered model predictors is small according to the criteria described in the Appendix C. The results of the provided table indicate that the model predictors considered do not have a significant effect on the variation of the dependent variable, and further, denote small effect size on the proportion of variance they are able to explain from the dependent variable. Therefore, and in line with the results obtained with ANCOVA, **we did not find a statistically significant association between the use of MS architecture and the improvement in the velocity of software projects**, nor in the case of the confounders considered.

Furthermore, we provide results for the achievement of the model assumptions with ANCOVA and Linear Regression accordingly in the replication package, as well as the illustration of model residuals, thus providing a visual model evaluation 7.

6 Discussion

This study aimed to empirically investigate the impact of microservice architecture on the development velocity of OSS projects. By adopting a cohort study methodology, we were able to systematically compare projects that used MS from the beginning of their developers with those that maintained a monolithic architecture. Our analysis included a comprehensive examination of 265 open source projects, with 58 MS-based systems and 207 monolithic ones.

The key findings of the study are as follows.

- Similar development velocity between groups: The performed analysis results demonstrated existing similarity between velocity observed in MS projects and non-microservices projects, as there was no statistically significant difference between the two groups' observations.
- Quality of data: Despite a substantial reduction in the number of analyzable projects due to stringent inclusion and exclusion criteria, the resulting data set maintained a high level of quality. This ensured that the observed effects were robust and not confounded by the most common extraneous variables.
- Small Effect Size and Statistical Power: The computed statistical power was 0.055, with a small effect size of 0.17. Although these figures are lower than initially expected, they still provide valuable insight into the real-world impact of MS on development velocity.
- Impact from confounders: The adjusted analysis did not provide show statistical significant impact from the included confounding variables. However, the resulting probabilities contained a small effect size.

6.1 Implications

Implications for Researchers The results of this study have several significant implications for researchers in the field of empirical software engineering. Firstly, this research stands as one of the first applications of cohort study methodology within this domain, demonstrating its feasibility and utility. The design of the cohort study allowed for a thorough investigation of the causal impact of MS on development velocity by systematically comparing two well-defined groups over a specified period. This approach provides high-level evidence, which is often difficult to achieve in observational studies.

The application of cohort studies in software engineering can inspire further empirical investigations into other architectural or methodological interventions. Researchers can leverage the detailed methodology described in this study to design their cohort studies, contributing to a more robust and evidence-based understanding of various software engineering practices.

Implications for Practitioners For industry practitioners, the findings offer empirical support for the widely held belief that MS adoption can positively impact the development velocity. This study specifically examined OSS projects over two years, ensuring a comprehensive analysis of active and mature projects. The results confirmed that projects employing microservice architectures indeed exhibit higher velocity in terms of issue resolution compared to their monolithic counterparts, confirming the initial assumption made by practitioners (Fowler and Lewis 2014; Newman 2015)

This empirical confirmation is crucial for practitioners considering the transition to MS. It provides a data-driven basis for expecting improvements in development speed, which is particularly valuable in fast-paced and competitive software development environments. The study also highlights the importance of maintaining a consistent microservice architecture throughout the project's lifecycle to achieve these benefits. However, we also believe that practitioners should not take our result, or the expected improvement in velocity, as the main driver to migrate to MS, but they should also consider other aspects before making decisions.

Comprehensive Investigation and Data Quality In this work, we performed an exhaustive examination of all open-source projects older than two years available in public repositories. The rigorous data collection and filtering process ensured the inclusion of high-quality projects, thus enhancing the validity of the findings. The trade-off between power and data quality, while challenging, ultimately resulted in a robust data set that allowed the derivation of meaningful insights.

Applicability of Cohort Studies The successful implementation of a cohort study in this context highlights its applicability in empirical software engineering. By adopting this methodology, researchers can obtain higher-level evidence on causal questions that are otherwise difficult to investigate without conducting controlled experiments. The detailed methodology and systematic approach used in this study can be used as a reference for future researchers.

7 Threats to Validity

Construct Validity The measurement of MS adoption could be inaccurate. As GitHub does not provide exact data for this, we rely on the number of Dockerfiles in the repository adopting

the same MS identification algorithm used by Amoroso d'Aragona et al. (2024). This threat of projects not using MS despite having Dockerfiles is attempted to mitigate by requiring at least three Dockerfiles and excluding volumes and similar files from the count. We selected MS projects from the list of manually labeled MS-based projects from the Amoroso d'Aragona et al. (2024) dataset. Additionally, we acknowledge that selecting a different length for the velocity calculation time window could alter the results. For example, the window might be inadequate if the calculation period has holidays. Moreover, projects evolve, with changes in confounding factors such as project size, team composition, and development practices. These evolving characteristics may have influenced our results, and we acknowledge them as potential threats to validity.

Internal Validity Cohort studies are sensitive to case and control groups not being comparable. We ensured their similarity by applying strict eligibility criteria and controlling for confounders using matching and statistical adjustment. Also, the data collection procedure is the same for all subjects as the projects were originally sampled from GitHub, which ensures that the data were logged similarly for all subjects and enables automated data collection. To further ensure the similarity of the projects, they were followed from their creation until they are two years old, and, therefore, they were in the same phase of their evolution. Naturally, no two projects are the same and it is not possible to prove they are similar enough for comparison. However, we believe that the actions described previously are enough to ensure that the compared projects are similar enough on characteristics that are relevant to the relationship studied.

The project velocity might differ for several reasons. This was addressed by including several external variables that could affect it. We are aware that our confounder selection might be subjective and that different researchers might have selected different confounders. However, MSR does not have a curated list of potential confounders to be considered for different outcomes. We have tried to systematically identify confounders and included eight confounding variables in the study. However, it is not possible to include all relevant variables. This could be due to the large number that would lead to overfitting, not having the data for a variable due to using historical data, or some variables not fitting the granularity of the study. However, we have included those we considered the most relevant and those that were measurable from the historical data. Other potential confounders could have been, for example, factors that affected the complexity of issues, testing practices, domain, and use of the CI pipeline. We are also aware that some confounders might not be measurable or were not included in the historical data used in the study. This is a limitation of the use of retrospective data. In case the results would have been significant, we would have performed a sensitivity analysis to assess the impact required by unknown or unmeasured variables to change the result.

External Validity The included projects are open-source projects from GitHub that reach a certain level of maturity within two years of their creation. Therefore, the results are generalizable to young and active open source projects, that is, this study does not apply to older projects with greater maturity. However, the actual generalizability of the results will depend on the data collected.

Conclusion Validity The planned data analysis follows the structure generally adopted in cohort studies. Moreover, our project selection process includes only Docker-based projects; therefore, even if we believe that different containerization technologies should not have a different impact on the velocity than Docker, non-Docker-based projects might perform differently in some aspects.

8 Conclusion

The presented work investigated the effect of the microservice architecture on the time to resolve the issues reported in the issue tracker (velocity). The study was designed as a retrospective cohort study, which compared young software projects using microservice architecture with monolithic projects. The results from the crude analysis indicated a certain similarity between microservices projects and monolithic projects in terms of velocity. The result was adjusted for several confounding variables, such as the size of the project and the language. However, the analysis did not find significant evidence that confounders or the usage of the MS architecture would explain the detected variation in velocity.

Despite the results leaving unanswered questions, the work is paving the way toward the adoption of the cohort study methodology in empirical software engineering research, as it is one of the first cohort studies conducted in the field. The study shows that the retrospective cohort study method can be applied in the software engineering domain and demonstrates its ability to collect data that provides higher confidence in the results obtained. The study also provides a subset of active and well maintained microservice-based projects.

For industry professionals, the findings offer empirical support for the common assumption that the adoption of MS can positively impact the development velocity.

However, more research is needed to identify and understand the specific factors that influence velocity, as they were not fully explored in this study. To address this, future research should focus on analyzing the impact of these evolving factors in more depth to better isolate the effect of MS adoption on velocity. Additionally, more experience is needed in applying the cohort study methodology in software engineering. Deeper insights into its implementation will help refine the approach and further encourage the software engineering community to adopt cohort studies for empirical research.

A Statistical methods

A.1 Commit and Issue Activity Trend Analysis

In the following appendix Section, we describe the performed trend analysis on the commit and issue activity data mined from the initially collected projects. As highlighted in Section 4, knowing the level of activity the considered projects had during the observational study is a crucial factor that affects the quality of the data. Otherwise, highly inactive projects could have a negative impact on the results and therefore enhance erroneous interpretations of the study results. Thus, we need to meticulously analyze the activity trend of the projects on commits and issues, as well as reasonably select the time frames to be used to calculate the trends to correctly monitor the development activity level of the considered projects and, therefore, their suitability to be analyzed in the study.

As version control activity is inherently stochastic, trend analysis was performed using aggregated data. Committing activity in short time frames, such as time periods *daily* or *weekly*, can cause the activity trends to present sharp fluctuations, which would not accurately describe the activity of a project. For example, a week without creating or closing issues does not mean that a project does not actively track its issues. Hence, we considered *monthly, bimonthly, quarterly* and *quadrimestral* time frames. In the registered report, we suggested inspecting *monthly* series (Saarimäki et al. 2023), however, lengthening the time window

allows us to perform a more flexible filtering of the projects given the sharp drop in the previous step.

The *overall* activity trends were determined using the *Mann-Kendall* (MK) non-parametric trend test or *Mann-Kendall's* τ. The MK test tests for the presence of a monotonic trend of a variable in chronological time series observations, as well as evidence of linearity in the observed trend (Mann 1945). Furthermore, it does not need to adjust for any specific distribution of the data, thus providing a flexible but still robust measurement of the trend. A detailed description of the test is provided in the Appendix A.2.

Furthermore, we implemented *Linear Regression* by calculating *Ordinary Least Squares* under the assumption of linearity (Bianchi et al. 1999) to observe the direction of the trend for those projects that would have proven to have a positive trend through the MK test. We fitted a Linear Regression model for each chronologically ordered data series and computed the value of the model parameter β_1:

$$Y = \beta_0 + \beta_1 X + e$$

where β_0 is the regression intercept, β_1 is the model parameter for the observed value and therefore the slope of the model trend, and e is the theoretical random error. We considered that the activity trend of the projects should show a non-negative slope to consider it as an active project.

A.2 Mann-Kendall Non-Parametric Trend Test

To perform the MK test we measured the difference (Δ) of the consecutive observations, and similarly assigned the integer values 1, -1, and 0 to the calculated positive, negative, and no-difference instances, respectively. Consequently, the value of the test statistic S is calculated as the sum of the integers assigned:

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sign(y_j - y_i)$$

where $sign(y_j - y_i)$ stands for the integer labeling already mentioned, and therefore $j > i$ are the consecutive observations in the chronologically ordered data n. When S is a large positive number, the latter measured values happen to be larger than the previous values, showing therefore an upward trend. When S is a large negative number, the latter measured values are usually smaller than the previous values, showing a downward trend. Similarly, if a small value of S is computed, it shows a non-consistent trend.

Consequently, the τ test statistic is computed:

$$\tau = \frac{S}{n(n-1)/2}$$

which provides a range of values from -1 to +1. The MK test defines the hypotheses in trend analysis as follows:

- H_0: There is no statistically significant evidence to prove the existence of a trend in the observed data.
- H_1: There is statistically significant evidence to prove the existence of a trend in the observed data.

Therefore, the null hypothesis is rejected when S and τ are significantly different from zero. If a significant trend is found, the rate of change and, therefore, the trend, can be

estimated through the Sen slope estimator (Helsel and Hirsch 1993):

$$\beta = median\left(\frac{y_j - y_i}{x_j - x_i}\right)$$

for all consecutive $j > i$ where $i = \{1, 2, \ldots, n-1\}$ and $j = \{2, \ldots, \ldots, n\}$. Thus, the Sen slope estimates the slope of the observations used to compute the test statistic S. Thus, we conduct the exclusion in the following way:

- H_0 is not rejected: The projects do not have a statistically significant trend in the observed data irrelevant to the direction of the trend.
- *Negative trend:* Projects that show a significant negative trend in their value S.
- *Negative slope:* Projects that show a negative slope in their trend.

A.3 Generalized Power Analysis through Monte Carlo Simulation

The expanded approach is based on the work performed by Du et al. (2017), further details, as well as applied examples, can be found in their original work. The following lines demonstrate the application of the Monte Carlo-based method in the context of the power analysis for *two sample t-test* as well as the steps performed through the adopted software in R.

The *two sample t-test* tests if two independent population means are equal, therefore defining the hypotheses below. The alternative can be two- or one-tailed; we present the second option given the context of our study.

$$H_0 : \mu_1 - \mu_2 = 0$$

$$H_1 : \mu_1 - \mu_2 > 0$$

The pooled-variance t-test where the statistic t_{pooled} is defined as follows:

$$t_{pooled} = \frac{\bar{y}_1 - \bar{y}_2}{\sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2}}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}$$

follows a distribution t with

$$n_1 + n_2 - 2$$

degrees of freedom. n_i are the sample sizes and $\bar{y}_i$ and s_i^2 accordingly for $i = 1, 2$. The t-test assumes *homogeneity* and *normality*. In such a scenario where the assumptions are violated, the Monte Carlo-based method could be used for power analysis. The steps for running this method for the calculation of the power of the t-test on two samples given sample sizes n_i are derived as follows:

1. Considering such population means mu_i that the defined null hypothesis H_0 is respected, standard deviations σ_i, skewness values γ_{1i} and kurtosis values γ_{1i} for $i = 1, 2$ given the definition formulas in Section 4.7, we generate sets of non-normal data R_0 with sample sizes n_1 and n_2 accordingly.
2. For each of the sets simulated R_0, we calculated the mean and variance for each group as $\bar{y}_{01j}$, $\bar{y}_{02j}$, s_{01j}^2, and s_{02j}^2 given

$$j = 1, ..., R_0$$

so that the values for the test statistics

$$t^*_{0j} = \frac{\bar{y}_{01j} - \bar{y}_{02j}}{\sqrt{\frac{s^2_{01j}}{n_1} + \frac{s^2_{02j}}{n_2}}}$$

are computed. Then, we obtain the critical value c_α according to the specified α value and the H_1.

3. Taking into account μ_{11} and μ_{21} as means of the two groups under H_1, we again simulate R_1 of non-normal data, following the same criteria used with R_0.
4. We repeat Step 2 to calculate the means and variances for the simulated sets R_1 and calculate the corresponding values for the t^*_{1i} t-statistic.
5. The power is estimated as the proportion that t^*_{ai} is greater than the critical value

$$c_\alpha : \pi = \#(t^*_{ai} > c_\alpha)/R_1$$

A.4 Balance Metrics for Matching Assessment

In the following lines, we provide the definitions of the confounder balance we used to assess the balance level after matching the data. Therefore, we used the following commonly used metrics to assess the balance (Ho et al. 2007):

– *Standardized Mean Differences* (SMD) is the difference in the means of each confounding variable between case groups standardized by the standardization factor so that it is on the same scale for all confounding variables (Stuart et al. 2013). SMDs indicate good balance when values close to 0 are achieved.

$$SMD = \frac{MD}{SD_{measured}}, where \quad SD_{measured} = \sqrt{\frac{SD^2_{cases} + SD^2_{controls}}{2}}$$

where *MD* is the Mean Difference of the confounding variable between groups, *SD* is the computed Standard Deviation.

– *Variance Ratio* (VRatio) is the ratio of the variance of a confounding variable in one group to that in the other. Variance ratios close to 1 indicate good balance since they imply the variances of the samples are similar (Austin 2009).

$$VRatio = \frac{S^2_{cases}}{S^2_{controls}}$$

where *S* is the sample variance for the population considered.

– *Empirical CDF statistics*: Empirical Cumulative Distribution Functions (eCDF) of each confounder between groups allow assessment of imbalance across the entire confounder distribution of that confounding variable. We calculate the mean eCDF difference (Austin 2009) and the maximum eCDF difference, better known as the Kolmogorov-Smirnov statistic (Austin and Stuart 2015).

$$\hat{F}_n(t) = \frac{1}{n+1} \sum_{i=1}^{n} 1_{X_i \leq t}$$

where F stands for the value of eCDF, n is the number of data points and t is fixed by the confounder distribution.

– *Visual Diagnostics* such as eCDF plots and kernel density plots to see how the confounder distributions differ from each other (Ho et al. 2007).

B Results from Assessing the Quality Of Matching

The initial state of the data before the match is presented in Table 12. Comparing these numbers with those presented in Table 6 shows that matching did not create large differences as stated in the main text.

To further investigate the quality and the effect of having 10 unmatched subjects, we visually inspected the behavior of the model predictors through their kernel distribution function and eCDF plots, illustrated in Figs. 13 and 14. Although there were visible cases of small imbalances between groups, there was no evidence of a lack of complete overlap in any of the continuous confounding variables displayed. Furthermore, there were no substantial

Table 12 Summary of the balance of all unmatched data

	MCa	MCo	SMD	VRatio	eCDF Mean	eCDF Max
Velocity Start	32.35	21.75	0.21	2.87	0.04	0.10
Project Size	27,075.29	34,812.02	-0.08	0.67	0.08	0.14
BASH	0.02	0.00	0.10	.	0.01	0.01
C*	0.00	0.01	-0.14	.	0.01	0.01
C #*	0.10	0.04	0.21	.	0.06	0.06
C++*	0.00	0.01	-0.14	.	0.01	0.01
ClojureScript*	0.00	0.00	-0.08	.	0.00	0.00
Elixir*	0.02	0.00	0.13	.	0.02	0.02
F #*	0.00	0.00	-0.08	.	0.00	0.00
Go*	0.07	0.13	-0.22	.	0.06	0.06
Java*	0.14	0.07	0.20	.	0.07	0.07
JavaScript*	0.19	0.17	0.04	.	0.02	0.02
PHP*	0.10	0.06	0.15	.	0.05	0.05
Python*	0.22	0.27	-0.10	.	0.04	0.04
Ruby*	0.02	0.08	-0.46	.	0.06	0.06
Rust*	0.03	0.01	0.14	.	0.02	0.02
Scala*	0.02	0.01	0.06	.	0.01	0.01
Shell*	0.02	0.03	-0.09	.	0.01	0.01
SQL*	0.00	0.00	-0.08	.	0.00	0.00
TypeScript	0.05	0.10	-0.20	.	0.04	0.04
# PL	3.40	3.10	0.14	1.84	0.02	0.06
CY= 2016	0.21	0.21	-0.00	.	0.00	0.00
CY= 2017	0.14	0.24	-0.30	.	0.10	0.10
CY= 2018	0.26	0.26	-0.01	.	0.00	0.00
CY= 2019	0.21	0.19	0.03	.	0.01	0.01
CY= 2020	0.19	0.10	0.24	.	0.09	0.0
# Commits	569.43	511.87	0.08	0.59	0.04	0.14
# Issues	186.67	193.27	-0.02	1.19	0.05	0.13
# Developers	11.81	9.42	0.15	2.02	0.03	0.12

* We report the main programming language PL = programming language, CY = creation year
(#: *Number of*, MCa: *Mean of the Cases*, MCo: *Mean of the Controls*, SMD: Standardized Mean Differences)

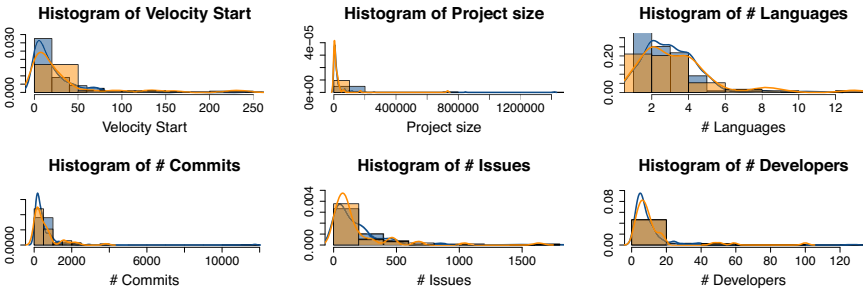

Fig. 13 Kernel density function plots of continuous confounders after Exact matching. (Blue: Controls, Orange: Cases)

changes in the categorical variable *Creation year* based on the distribution shape illustrated previously (its illustration is available in the replication package).

C On the Effect-Size Estimation Statistics

C.1 Effect Size Statistics for the ANCOVA Model

- **Eta-Squared** (η^2)**:** Explains how much of the variation in the dependent variable can bu predicted (or explained by) the variation on the adjusted variable (Olejnik and Algina 2003). Once the model is built, the η^2 statistic is computed by measuring the **percent** of the total variance through the resulting sum-of-squares (SS) associated to the model predictor and the unexplained part, that is the residuals from the model. Thus, the η^2 is defined as follows:

$$\eta^2 = \frac{SS_{effect}}{SS_{total}} = \frac{SS_{effect}}{SS_{effect} + SS_{residuals}} \quad (1)$$

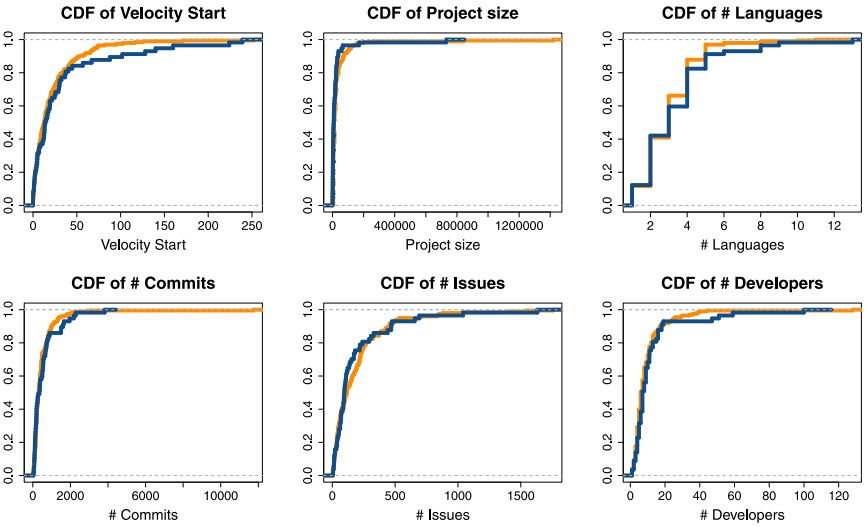

Fig. 14 eCDF plots of continuous confounders after Exact matching. (Blue: Controls, Orange: Cases)

- **Partial Eta-Squared (Partial η^2):** When more terms are added in the estimated model, the effect-size quantification can be expanded to capture how much variance is accounted by the concerning predictor in *total* and how much is accounted while *adjusting* for any other predictor too. While the first scenario is already covered by the η^2 statistic, the second one can be quantified through the Partial η^2 statistic (Olejnik and Algina 2003), presented as the following **percentage**:

$$Partial\ \eta^2 = \frac{SS_{effect}}{SS_{effect} + SS_{error}} \tag{2}$$

- **Partial Omega-Squared (Partial ω^2):** Looking into the perspective of unbiased effect size estimators, ω^2 stands as one of the most common utilized effect size statistics (Carroll and Nordholm 1975), and therefore, and an unbiased alternative to the η^2.

$$Partial\ \omega^2 = \frac{SS_{effect} - df_{effect} MS_{s/Cells}}{SS_{effect} + (N - df_{effect}) MS_{s/Cells}} \tag{3}$$

where:

  - SS_{effect} is the sum of squares computed from the concerning predictor.
  - $MS_{s/Cells}$ is the residual variance or the *unexplained variability* after adjusting analysis for the confounders.
  - N is the total population size.
  - df_{effect} accounts for the degrees of freedom of each model predictor.

In this study, the interpretation of the effect size estimates is done through the Field's effect size rule (Field 2024):

  - Effect Size $<$ 0.01 - Very small
  - 0.01 $\leq$ Effect Size $<$ 0.06 - Small
  - 0.06 $\leq$ Effect Size $<$ 0.14 - Medium
  - Effect Size $\geq$ 0.14 - Large

- **Partial Epsilon-Squared (Partial ϵ^2):** Also referred as the *Adjusted* Eta-Squared (Allen 2017), it is presented as a less biased approach compared to the Partial ω^2 (Carroll and Nordholm 1975). Its' mathematical definition slightly differs from the one defining the Partial ω^2:

$$Partial\ \epsilon^2 = \frac{SS_{effect} - df_{effect} MS_{s/Cells}}{SS_{total}} \tag{4}$$

where:

  - SS_{effect} is the sum of squares computed from the concerning predictor.
  - $MS_{s/Cells}$ is the residual variance or the *unexplained variability* after adjusting analysis for the confounders.
  - SS_{totsl} is the total sum of squares.
  - df_{effect} accounts for the degrees of freedom of each model predictor.

Similarly, the interpretation of the effect size estimates is done through the Field's effect size rule (Field 2024).

C.2 Effect Size statistics for the Linear Regression model

- **Coefficient of Determination (R^2):** is the proportion of the variation in the dependent variable that is predictable from the independent variable(s) based on the fitted model (Agresti 2015). R^2 defines the *goodness-of-fit* of a model as follows:

$$R^2 = 1 - \frac{SS_{residuals}}{SS_{total}} \tag{5}$$

  where:

  - $SS_{residuals}$ denotes the sum of squares of the resulting model residuals.
  - SS_{total} is the total sum of squares.

  The better the linear regression fits the data in comparison to the simple average, the closer the value of R2 is to 1, thus denoting optimal model goodness-of-fit.
- **Adjusted R^2:** Stands as an attempt to account for the phenomenon of the R^2 automatically increasing when extra explanatory variables are added to the model (Agresti 2015). Thus, the adjusted version of the coefficient of determination penalizes the estimate on the increase of model predictors:

$$Adj\ R^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{6}$$

  where:

  - n is the population size.
  - p is the total number of explanatory variables in the model (excluding the intercept).

  The interpretation of the Adjusted R^2 stands as the same as the for the coefficient of determination.
- **Partial r-squared (Partial r^2):** Describes how much of the residual variance of the outcome (after partialing out the other covariates) a covariate explains (Chin et al. 1998). Hence, this statistic describes what **percentage** of the dependent variable's variability is explained by the rest of model predictors while keeping the concerning predictor aside. The estimation can be defined as follows:

$$Partial\ r^2 = \frac{SS_{resid,reduced} - SS_{resid,total}}{SS_{resid,reduced}} \tag{7}$$

  where:

  - $SS_{resid,reduced}$ is the residual sum of squares from the reduced model.
  - $SS_{resid,total}$ is the residual sum of squares from the total model, that is, the one with all the model predictors.

- **Partial f-squared (f^2):** Stands as a common measure of effect size (a transformation of the partial R2) that can also be used directly for sensitivity analysis (Cohen 2013). It is appropriate for calculating the effect size within a multiple regression model in which the independent variable of interest and the dependent variable are both continuous.

$$Partial\ f^2 = \frac{R^2}{1 - R^2} \tag{8}$$

  Thus, the effect size is defined within the following value ranges:

  - $0.02 \leq$ Effect Size < 0.15 - Small

  - $0.15 \leq$ Effect Size < 0.35 - Medium
  - Effect Size ≥ 0.35 - Large

- **Partial f:** Defined as a derivation of the previous effect size statistic, the partial f estimates the unique contribution of the model predictor to the overall effect size of the model (Cinelli and Hazlett 2020). It's definition is presented as follows:

$$Partial\ f = \sqrt{\frac{R^2}{1 - R^2}} \tag{9}$$

  Thus, the effect size is defined within the following value ranges:

  - $0.10 \leq$ Effect Size < 0.25 - Small
  - $0.25 \leq$ Effect Size < 0.40 - Medium
  - Effect Size ≥ 0.40 - Large

Author Contributions We specify our contributions according to the CRediT taxonomy: Nyyti Saarimaki: Methodology, Formal Analysis, Writing - Original Draft Preparation. Mikel Robredo: Methodology, Writing - Original Draft Preparation. Valentina Lenarduzzi: Conceptualization, Methodology, Supervision. Sira Vegas: Conceptualization, Methodology, Supervision. Natalia Juristo: Conceptualization, Methodology, Supervision. Davide Taibi: Conceptualization, Methodology, Supervision.

Funding Open Access funding provided by University of Oulu (including Oulu University Hospital). This research was supported by grant PID2022-137846NB-I00 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe".
This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), grant reference C22/IS/17373407/LOGODOR. For the purpose of open access, and in fulfillment of the obligations arising from the grant agreement, the authors have applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript Version arising from this submission.

Data Availability We include all raw data in the replication package as reported in Section 4.9.

Declarations

Ethical Approval The paper did not need ethical approval.

Informed Consent The paper did not need informed consent.

Conflict of Interest We declare that we have no competing interests.

References

Agresti A (2015) Foundations of linear and generalized linear models. John Wiley & Sons
Allen R (2017) Statistics and experimental design for psychologists: A model comparison approach. World Scientific Publishing Company

Amoroso d'Aragona D, Bakhtin A, Li X, Su R, Adams L, Aponte E, Boyle F, Boyle P, Koerner R, Lee J, Tian F, Wang Y, Nyyssölä J, Quevedo E, Md Rahaman S, Abdelfattah AS, Mäntylä M, Cerny T, Taibi D (2024) A dataset of microservices-based open-source projects. International conference on mining software repositories (MSR) 21:504–509
Austin PC (2009) Balance diagnostics for comparing the distribution of baseline covariates between treatment groups in propensity-score matched samples. Stat Med 28(25):3083–3107
Austin PC, Stuart EA (2015) Moving towards best practice when using inverse probability of treatment weighting (iptw) using the propensity score to estimate causal treatment effects in observational studies. Stat Med 34(28):3661–3679
Ayas HM, Leitner P, Hebig R (2023) An empirical study of the systemic and technical migration towards microservices. Empir Softw Eng 28(4):85. https://doi.org/10.1007/S10664-023-10308-9
Benson K, Hartz AJ (2000) A comparison of observational studies and randomized, controlled trials. N Engl J Med 342(25):1878–1886
Bianchi M, Boyle M, Hollingsworth D (1999) A comparison of methods for trend estimation. Appl Econ Lett 6(2):103–109
Blair RC, Higgins JJ (1980) A comparison of the power of wilcoxon's rank-sum statistic to that of student's t statistic under various nonnormal distributions. J Educ Stat 5(4):309–335
Box GE, Cox DR (1964) An analysis of transformations. J Royal Stat Soc Ser B Stat Methodol 26(2):211–243
Boyter B (2018) Sloc cloc and code (scc). https://github.com/boyter/scc
Carroll RM, Nordholm LA (1975) Sampling characteristics of kelley's ε and hays' ω. Educ Psychol Meas 35(3):541–554
Casella G, Berger R (2024) Statistical inference. CRC Press
Chin WW et al (1998) The partial least squares approach to structural equation modeling. Mod Methods Bus Res 295(2):295–336
Cinelli C, Hazlett C (2020) Making sense of sensitivity: Extending omitted variable bias. J Royal Stat Soc Ser B Stat Methodol 82(1):39–67
Cohen J (2013) Statistical power analysis for the behavioral sciences. Routledge
Concato J, Shah N, Horwitz RI (2000) Randomized, controlled trials, observational studies, and the hierarchy of research designs. N Engl J Med 342(25):1887–1892
D'Agostino RB (2017) Goodness-of-fit-techniques. Routledge
Di Cicco ME, Ragazzo V, Jacinto T (2016) Mortality in relation to smoking: the british doctors study. Breathe 12:275–276
Dragoni N, Giallorenzo S, Lafuente AL, Mazzara M, Montesi F, Mustafin R, Safina L (2017) Microservices: yesterday, today, and tomorrow. In: Present and ulterior software engineering, Springer, pp 195–216
Dragoni N, Lanese I, Larsen ST, Mazzara M, Mustafin R, Safina L (2018) Microservices: How to make your application scale. In: Petrenko AK, Voronkov A (eds) Perspectives of system informatics, springer international publishing, Cham, pp 95–104
Du H, Zhang Z, Yuan KH (2016) (2017) Power analysis for t-test with non-normal data and unequal variances. Quantitative psychology: The 81st annual meeting of the psychometric society, Asheville, North Carolina. Springer, pp 373–380
Ellis PD (2010) The essential guide to effect sizes: Statistical power, meta-analysis, and the interpretation of research results. Cambridge University Press
Faresjö T, Faresjö Å (2010) To match or not to match in epidemiological studies–same outcome but less power. Int J Environ Res Public Health 7(1):325–332
Field A (2024) Discovering statistics using IBM SPSS statistics. Sage publications limited
Fowler M, Lewis J (2014) Microservices a definition of this new architectural term. https://bit.ly/3zk5xXr
Gordis L (2014) Epidemiology, fifth, edition. Elsevier Saunders, Philadelphia
Grimes DA, Schulz KF (2002) Cohort studies: marching towards outcomes. Lancet 359(9303):341–345
Grimes DA, Schulz KF (2002) Descriptive studies: what they can and cannot do. Lancet 359(9301):145–149
Hart A (2001) Mann-whitney test is not just a test of medians: differences in spread can be important. Bmj 323(7309):391–393
Helsel DR, Hirsch RM (1993) Statistical methods in water resources. Elsevier
Ho DE, Imai K, King G, Stuart EA (2007) Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. Polit Anal 15(3):199–236
Hollander M, Wolfe DA, Chicken E (2013) Nonparametric statistical methods. John Wiley & Sons
Inc D (2023) Dockerfile reference. https://docs.docker.com/engine/reference/builder/
Jamshidi P, Pahl C, Mendonça NC, Lewis J, Tilkov S (2018) Microservices: The journey so far and challenges ahead. IEEE Software 35(3):24–35
Jangla K (2018a) Accelerating Development Velocity Using Docker: Docker Across Microservices. apress
Jangla K (2018b) Accelerating Development Velocity Using Docker: Docker Across Microservices. Apress

Lenarduzzi V, Lomio F, Saarimäki N, Taibi D (2020) Does migrating a monolithic system to microservices decrease the technical debt? J Syst Softw 169:110710
Lomio F, Codabux Z, Birtch D, Hopkins D, Taibi D (2022) On the benefits of the accelerate metrics: An industrial survey at vendasta. In: 2022 IEEE international conference on software analysis, evolution and reengineering (SANER), pp 46–50, https://doi.org/10.1109/SANER53432.2022.00017
Ma Y, Dey T, Bogart C, Amreen S, Valiev M, Tutko A, Kennard D, Zaretzki R, Mockus A (2021) World of code: enabling a research workflow for mining and analyzing the universe of open source vcs data. Empir Softw Eng 26:1–42
Mann HB (1945) Nonparametric tests against trend. Econometrica: Journal of the econometric society, pp 245–259
Moore DS, McCabe GP (1989) Introduction to the practice of statistics. WH Freeman/Times Books/Henry Holt & Co
Morabia A (2004) A history of epidemiologic methods and concepts. vol 1
Newman S (2015) Building Microservices: Designing Fine-Grained Systems, 1st edn. O'Reilly Media
Newman S (2021) Building microservices: designing fine-grained systems. " O'Reilly Media, Inc."
O'brien RM (2007) A caution regarding rules of thumb for variance inflation factors. Quality & quantity 41:673–690
Olejnik S, Algina J (2003) Generalized eta and omega squared statistics: measures of effect size for some common research designs. Psychol Methods 8(4):434
Oppewal H (2010) Concept of causality and conditions for causality. Wiley International encyclopedia of marketing
Pahl C, Jamshidi P, Zimmermann O (2018) Architectural principles for cloud software. ACM Trans Internet Technol 18(2):Article No.: 17
Razali NM, Wah YB et al (2011) Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. J Stat Model Anal 2(1):21–33
Rosenbaum P (2002) Observational studies, 2nd edn springer. New York, New York, USA[Google Scholar]
Rosenbaum PR (2010) Design of Observational Studies, 1st edn. Springer Series in Statistics, Springer-Verlag, New York, NY
Royston JP (1982) An extension of shapiro and wilk's w test for normality to large samples. J Royal Stat Soc Ser C (Appl Stat) 31(2):115–124
Rubin DB (2001) Using propensity scores to help design observational studies: application to the tobacco litigation. Health Serv Outcomes Res Methodol 2:169–188
Saarimäki N, Lenarduzzi V, Vegas S, Juristo N, Taibi D (2020) Cohort studies in software engineering: A vision of the future. In: Proceedings of the 14th ACM/IEEE international symposium on empirical software engineering and measurement (ESEM), pp 1–6
Saarimäki N, Manero MR, Juristo N, Taibi D, Lenarduzzi V, et al. (2023) Does microservices adoption impact the development velocity? a cohort study. a registered report. arXiv preprint arXiv:2306.02034
Sheather S (2009) A modern approach to regression with R. Springer Science & Business Media
Soldani J, Tamburri DA, Heuvel WJVD (2018) The pains and gains of microservices: A systematic grey literature review. J Syst Softw 146:215–232
Song JW, Chung KC (2010) Observational studies: cohort and case-control studies. Plast Reconstr Surg 126(6):2234
Stuart EA, King G, Imai K, Ho D (2011) Matchit: nonparametric preprocessing for parametric causal inference. J Stat Softw
Stuart EA, Lee BK, Leacy FP (2013) Prognostic score-based balance measures can be a useful diagnostic for propensity score methods in comparative effectiveness research. J clin Epidemiol 66(8):S84–S90
Taibi D, Lenarduzzi V, Pahl C (2017) Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. IEEE Cloud Comput 4(5):22–32
Trichopoulos D, Kalandidi A, Sparros L, Macmahon B (1981) Lung cancer and passive smoking. Int J Cancer 27(1):1–4. https://doi.org/10.1002/ijc.2910270102
VanderWeele TJ (2019) Principles of confounder selection. Eur J Epidemiol 34:211–219
Von Elm E, Altman DG, Egger M, Pocock SJ, Gøtzsche PC, Vandenbroucke JP (2007) The strengthening the reporting of observational studies in epidemiology (strobe) statement: guidelines for reporting observational studies. Lancet 370(9596):1453–1457
Waseem M, Liang P, Márquez G, Salle AD (2020) Testing microservices architecture-based applications: A systematic mapping study. In: 2020 27th asia-pacific software engineering conference (APSEC), pp 119–128, https://doi.org/10.1109/APSEC51365.2020.00020
Wu M, Zhang Y, Liu J, Wang S, Zhang Z, Xia§X, Mao X (2022) On the way to microservices: Exploring problems and solutions from online q&a community. In: 2022 IEEE international conference on software

analysis, evolution and reengineering (SANER), pp 432–443, https://doi.org/10.1109/SANER53432.2022.00058
Zhang Z (2014) Monte carlo based statistical power analysis for mediation models: Methods and software. Behav Res Methods 46:1184–1198

Authors and Affiliations

Nyyti Saarimäki[1] · Mikel Robredo[2] · Valentina Lenarduzzi[2] · Sira Vegas[3] · Natalia Juristo[3] · Davide Taibi[2]

✉ Valentina Lenarduzzi
valentina.lenarduzzi@oulu.fi

Nyyti Saarimäki
nyyti.saarimaki@uni.lu

Mikel Robredo
mikel.robredomanero@oulu.fi

Sira Vegas
sira.vegas@upm.es

Natalia Juristo
natalia@fi.upm.es

Davide Taibi
davide.taibi@oulu.fi

1 University of Luxembourg, Luxembourg, Luxembourg

2 University of Oulu, Oulu, Finland

3 Universidad Politécnica de Madrid, Madrid, Spain