hw3

1.Study Summary

1.1 Overview

We collected totally 45 experimental studies papers, which utilize the methodology of case study, published in 2020, including 9 from ESEM, 30 from EMSE and 6 from EASE. After our further study, we discovered that the paper "Case Survey Studies in Software Engineering Research" from ESEM was classified improperly because the major research method of this paper is literature review in spite of its title. We've made corresponding adjustment taking our discovery into consideration.

表1: ESEM

ID	CITATION
1	Gouri Deshpande and Guenther Ruhe. 2020. Beyond Accuracy: ROI–driven Data Analytics of Empirical Data. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20). Association for Computing Machinery, New York, NY, USA, Article 37, 1—6. DOI:https://doi.org/10.1145/3382494.3422159
2	Héctor Cadavid, Vasilios Andrikopoulos, Paris Avgeriou, and John Klein. 2020. A Survey on the Interplay between Software Engineering and Systems Engineering during SoS Architecting. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20). Association for Computing Machinery, New York, NY, USA, Article 2, 1—11. DOI:https://doi.org/10.1145/3382494.3410671
3	Inger Anne Tøndel, Daniela Soares Cruzes, and Martin Gilje Jaatun. 2020. Using Situational and Narrative Analysis for Investigating the Messiness of Software Security. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20). Association for Computing Machinery, New York, NY, USA, Article 27, 1—6. DOI:https://doi.org/10.1145/3382494.3422162
4	Hugo Jernberg, Per Runeson, and Emelie Engström. 2020. Getting Started with Chaos Engineering – design of an implementation framework in practice. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20). Association for Computing Machinery, New York, NY, USA, Article 43, 1—10. DOI:https://doi.org/10.1145/3382494.3421464

5	Valentina Lenarduzzi, Vladimir Mandić, Andrej Katin, and Davide Taibi. 2020. How long do Junior Developers take to Remove Technical Debt Items? In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20). Association for Computing Machinery, New York, NY, USA, Article 30, 1—6. DOI:https://doi.org/10.1145/3382494.3422169
6	Zakaria Ournani, Romain Rouvoy, Pierre Rust, and Joel Penhoat. 2020. On Reducing the Energy Consumption of Software: From Hurdles to Requirements. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20). Association for Computing Machinery, New York, NY, USA, Article 14, 1—12. DOI:https://doi.org/10.1145/3382494.3410678
7	Ehsan Firouzi, Ashkan Sami, Foutse Khomh, and Gias Uddin. 2020. On the use of C# Unsafe Code Context: An Empirical Study of Stack Overflow. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20). Association for Computing Machinery, New York, NY, USA, Article 39, 1—6. DOI:https://doi.org/10.1145/3382494.3422165
8	Kamonphop Srisopha, Daniel Link, Devendra Swami, and Barry Boehm. 2020. Learning Features that Predict Developer Responses for iOS App Store Reviews. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20). Association for Computing Machinery, New York, NY, USA, Article 12, 1—11. DOI:https://doi.org/10.1145/3382494.3410686
9	Carmen Coviello, Simone Romano, Giuseppe Scanniello, and Giuliano Antoniol. 2020. GASSER: Genetic Algorithm for teSt Suite Reduction. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20). Association for Computing Machinery, New York, NY, USA, Article 36, 1—6. DOI:https://doi.org/10.1145/3382494.3422157

表2: EASE

ID	CITATION
1	Mansooreh Zahedi, Roshan Namal Rajapakse, and Muhammad Ali Babar. 2020. Mining Questions Asked about Continuous Software Engineering: A Case Study of Stack Overflow. In Proceedings of the Evaluation and Assessment in Software Engineering (EASE '20). Association for Computing Machinery, New York, NY, USA, 41—50. DOI:https://doi.org/10.1145/3383219.3383224
2	Zi Peng, Jinqiu Yang, Tse-Hsun (Peter) Chen, and Lei Ma. 2020. A first look at the integration of machine learning models in complex autonomous driving systems: a case study on Apollo. Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Association for Computing Machinery, New York, NY, USA, 1240—1250. DOI:https://doi.org/10.1145/3368089.3417063
3	Babak A. Farshchian, Hilde Sætertrø, and Marianne Stålaker. 2020. Experiences from an Interpretative Case Study of Innovative Public Procurement of Digital Systems in the Norwegian Public Sector. In Proceedings of the Evaluation and Assessment in Software Engineering (EASE '20). Association for Computing Machinery, New York, NY, USA, 373—374. https://doi.org/10.1145/3383219.3383271
4	Anh Nguyen-Duc, Ingrid Sundbø, Elizamary Nascimento, Tayana Conte, Iftekhar Ahmed, and Pekka Abrahamsson. 2020. A Multiple Case Study of Artificial Intelligent System Development in Industry. In Proceedings of the Evaluation and Assessment in Software Engineering (EASE '20). Association for Computing Machinery, New York, NY, USA, 1—10. https://doi.org/10.1145/3383219.3383220
5	Shanshan Li, Qianwen Xu, Peiyu Hou, Xiudi Chen, Yanze Wang, He Zhang, and Guoping Rong. 2020. Exploring the Challenges of Developing and Operating Consortium Blockchains: A Case Study. In Proceedings of the Evaluation and Assessment in Software Engineering (EASE '20). Association for Computing Machinery, New York, NY, USA, 398—404. https://doi.org/10.1145/3383219.3383276
6	Nils Brede Moe, Viktoria Stray, and Marcus R. Goplen. 2020. Studying Onboarding in Distributed Software Teams: A Case Study and Guidelines. In Proceedings of the Evaluation and Assessment in Software Engineering (EASE '20). Association for Computing Machinery, New York, NY, USA, 150—159. https://doi.org/10.1145/3383219.3383235

表3: EMSE

ID	CITATION

1	Oliveira, E., Fernandes, E., Steinmacher, I. *et al.* Code and commit metrics of developer productivity: a study on team leaders perceptions. *Empir Software Eng* **25,** 2519—2549 (2020). https://doi.org/10.1007/s10664–020–09820–z
2	Razzaq, A., Le Gear, A., Exton, C. *et al.* An empirical assessment of baseline feature location techniques. *Empir Software Eng* **25,** 266—321 (2020). https://doi.org/10.1007/s10664–019–09734–5
3	Lee, D., Rajbahadur, G.K., Lin, D. *et al.* An empirical study of the characteristics of popular Minecraft mods. *Empir Software Eng* **25,** 3396—3429 (2020). https://doi.org/10.1007/s10664–020–09840–9
4	Lee, D., Lin, D., Bezemer, CP. *et al.* Building the perfect game — an empirical study of game modifications. *Empir Software Eng* **25,** 2485—2518 (2020). https://doi.org/10.1007/s10664–019–09783–w
5	Lee, D., Lin, D., Bezemer, CP. *et al.* Building the perfect game — an empirical study of game modifications. *Empir Software Eng* **25,** 2485—2518 (2020). https://doi.org/10.1007/s10664–019–09783–w
6	Morasca, S., Lavazza, L. On the assessment of software defect prediction models via ROC curves. *Empir Software Eng* **25,** 3977—4019 (2020). https://doi.org/10.1007/s10664–020–09861–4
7	Said, W., Quante, J. & Koschke, R. Mining understandable state machine models from embedded code. *Empir Software Eng* **25,** 4759—4804 (2020). https://doi.org/10.1007/s10664–020–09865–0
8	Vassallo, C., Proksch, S., Zemp, T. *et al.* Every build you break: developer–oriented assistance for build failure resolution. *Empir Software Eng* **25,** 2218—2257 (2020). https://doi.org/10.1007/s10664–019–09765–y
9	Rodríguez–Pérez, G., Robles, G., Serebrenik, A. *et al.* How bugs are born: a model to identify how bugs are introduced in software components. *Empir Software Eng* **25,** 1294—1340 (2020). https://doi.org/10.1007/s10664–019–09781–y
10	Aktas, E.U., Yilmaz, C. Automated issue assignment: results and insights from an industrial case. *Empir Software Eng* **25,** 3544—3589 (2020). https://doi.org/10.1007/s10664–020–09846–3
11	Zhou, J., Wang, S., Bezemer, CP. *et al.* Bounties on technical Q&A sites: a case study of Stack Overflow bounties. *Empir Software Eng* **25,** 139—177 (2020). https://doi.org/10.1007/s10664–019–09744–3
12	Heeager, L.T., Nielsen, P.A. Meshing agile and plan–driven development in safety–critical software: a case study. *Empir Software Eng* **25,** 1035—1062 (2020). https://doi.org/10.1007/s10664–020–09804–z
13	Sharma, T., Singh, P. & Spinellis, D. An empirical investigation on the relationship between design and architecture smells. *Empir Software Eng* **25,** 4020—4068 (2020). https://doi.org/10.1007/s10664–020–09847–2

14	Cotroneo, D., Iannillo, A.K., Natella, R. *et al.* A comprehensive study on software aging across android versions and vendors. *Empir Software Eng* **25,** 3357—3395 (2020). https://doi.org/10.1007/s10664–020–09838–3
15	Marques, R., Costa, G., Mira da Silva, M. *et al.* A gamification solution for improving Scrum adoption. *Empir Software Eng* **25,** 2583—2629 (2020). https://doi.org/10.1007/s10664–020–09816–9
16	Trautsch, A., Herbold, S. & Grabowski, J. A longitudinal study of static analysis warning evolution and the effects of PMD on software quality in Apache open source projects.*Empir Software Eng* **25,** 5137—5192 (2020). https://doi.org/10.1007/s10664–020–09880–1
17	Kondo, M., Oliva, G.A., Jiang, Z.M.(. *et al.* Code cloning in smart contracts: a case study on verified contracts from the Ethereum blockchain platform. *Empir Software Eng* **25,** 4617—4675 (2020). https://doi.org/10.1007/s10664–020–09852–5
18	Li, S., Niu, X., Jia, Z. *et al.* Guiding log revisions by learning from software evolution history. *Empir Software Eng* **25,** 2302—2340 (2020). https://doi.org/10.1007/s10664–019–09757–y
19	Higo, Y., Hayashi, S., Hata, H. *et al.* Ammonia: an approach for deriving project–specific bug patterns. *Empir Software Eng* **25,** 1951—1979 (2020). https://doi.org/10.1007/s10664–020–09807–w
20	Piantadosi, V., Fierro, F., Scalabrino, S. *et al.* How does code readability change during software evolution?. *Empir Software Eng* **25,** 5374—5412 (2020). https://doi.org/10.1007/s10664–020–09886–9
21	Das, T., Di Penta, M. & Malavolta, I. Characterizing the evolution of statically–detectable performance issues of Android apps. *Empir Software Eng* **25,** 2748—2808 (2020). https://doi.org/10.1007/s10664–019–09798–3
22	Ghanavati, M., Costa, D., Seboek, J. *et al.* Memory and resource leak defects and their repairs in Java projects. *Empir Software Eng* **25,** 678—718 (2020). https://doi.org/10.1007/s10664–019–09731–8
23	Amanatidis, T., Mittas, N., Moschou, A. *et al.* Evaluating the agreement among technical debt measurement tools: building an empirical benchmark of technical debt liabilities. *Empir Software Eng* **25,** 4161—4204 (2020). https://doi.org/10.1007/s10664–020–09869–w
24	ao, K., de Pádua, G.B., Shang, W. *et al.* Log4Perf: suggesting and updating logging locations for web–based systems' performance monitoring. *Empir Software Eng* **25,** 488—531 (2020). https://doi.org/10.1007/s10664–019–09748–z
25	Hunsen, C., Siegmund, J. & Apel, S. On the fulfillment of coordination requirements in open–source software projects: An exploratory study. *Empir Software Eng* **25,** 4379—4426 (2020).

	https://doi.org/10.1007/s10664–020–09833–8
26	Linåker, J., Regnell, B. What to share, when, and where: balancing the objectives and complexities of open source software contributions. *Empir Software Eng* **25,** 3799—3840 (2020). https://doi.org/10.1007/s10664–020–09855–2
27	Ahasanuzzaman, M., Hassan, S., Bezemer, CP. *et al.* A longitudinal study of popular ad libraries in the Google Play Store. *Empir Software Eng* **25,** 824—858 (2020). https://doi.org/10.1007/s10664–019–09766–x
28	Robert, C., Sotiropoulos, T., Waeselynck, H. *et al.* The virtual lands of Oz: testing an agribot in simulation. *Empir Software Eng* **25,** 2025—2054 (2020). https://doi.org/10.1007/s10664–020–09800–3
29	Kondo, M., German, D.M., Mizuno, O. *et al.* The impact of context metrics on just–in–time defect prediction. *Empir Software Eng* **25,** 890—939 (2020). https://doi.org/10.1007/s10664–019–09736–3
30	Oliva, G.A., Hassan, A.E. & Jiang, Z.M.(. An exploratory study of smart contracts in the Ethereum blockchain platform. *Empir Software Eng* **25,** 1864—1904 (2020). https://doi.org/10.1007/s10664–019–09796–5

1.2 Our target papers

We choose 5 papers for further analysis and comparison.

1.2.1 Automated issue assignment: results and insights from an industrial case (EMSE)

BackGround: We automate the process of assigning issue reports to development teams by using data mining approaches and share our experience gained by deploying the resulting system, called IssueTAG, at Softtech.

Aim: Being a subsidiary of the largest private bank in Turkey, Softtech on average receives 350 issue reports daily from the field, which need to be handled with utmost importance and urgency.

Method: We first empirically determine the data mining approach to be used in IssueTAG. We then deploy IssueTAG and make a number of valuable observations.

Result: IssueTAG has been making all the issue assignments at Softtech since its deployment on Jan 12, 2018.

Conclusion: Deploying IssueTAG presented us not only with an unprecedented opportunity to observe the practical effects of automated issue assignment, but also with an opportunity to carry out user studies, both of which (to the

best of our knowledge) have not been done before in this context.

1.2.2 On the impact of using trivial packages: an empirical case study on npm and PyPI（EMSE）

BackGround: Code reuse has traditionally been encouraged since it enables one to avoid re–inventing the wheel. Due to the *npm* left–pad package incident where a trivial package led to the breakdown of some of the most popular web applications such as Facebook and Netflix, some questioned such reuse. Reuse of trivial packages is particularly prevalent in platforms such as *npm*. To date, there is no study that examines the reason why developers reuse trivial packages other than in *npm*.

Aim: This paper has extended our previous work to strengthen the empirical evidence on the use of trivial packages by replicating and extending our study on the Python Package Index.

Method: We study two large platforms *npm* and *PyPI*. We mine more than 500,000 *npm* packages and 38,000 JavaScript applications and more than 63,000 *PyPI* packages and 14,000 Python applications to study the prevalence of trivial packages. We found that trivial packages are common, making up between 16.0% to 10.5% of the studied platforms. We performed surveys with 125 developers who use trivial packages to understand the reasons and drawbacks of their use.

Result: Our surveys revealed that trivial packages are used because they are perceived to be well implemented and tested pieces of code. However, developers are concerned about maintaining and the risks of breakages due to the extra dependencies trivial packages introduce. To objectively verify the survey results, we validate the most cited reason and drawback.

Conclusion: We find that contrary to developers' beliefs only around 28% of *npm* and 49% *PyPI* trivial packages have tests. However, trivial packages appear to be 'deployment tested' and to have similar test, usage and community interest as non–trivial packages. On the other hand, we found that 18.4% and 2.9% of the studied trivial packages have more than 20 dependencies in *npm* and *PyPI*, respectively.

1.2.3 Code cloning in smart contracts: a case study on verified contracts from the Ethereum blockchain platform(EMSE)

BackGround: Ethereum is a blockchain platform that hosts and executes smart contracts. Smart contracts have been used to implement cryptocurrencies and crowdfunding initiatives (ICOs). A major concern in Ethereum is the security of smart contracts. Different from traditional software development, smart contracts are immutable once deployed. Hence, vulnerabilities and bugs in smart contracts can lead to catastrophic financial loses.

Aim: In order to avoid taking the risk of writing buggy code, smart contract developers are encouraged to reuse pieces of code from reputable sources (e.g., OpenZeppelin). In this paper, we study code cloning in Ethereum. Our

goal is to quantify the amount of clones in Ethereum (RQ1), understand key characteristics of clone clusters (RQ2), and determine whether smart contracts contain pieces of code that are identical to those published by OpenZeppelin (RQ3).

Method: We applied Deckard, a tree–based clone detector, to all Ethereum contracts for which the source code was available.

Result: We observe that developers frequently clone contracts. In particular, 79.2% of the studied contracts are clones and we note an upward trend in the number of cloned contracts per quarter. With regards to the characteristics of clone clusters, we observe that: (i) 9 out of the top–10 largest clone clusters are token managers, (ii) most of the activity of a cluster tends to be concentrated on a few contracts, and (iii) contracts in a cluster to be created by several authors. Finally, we note that the studied contracts have different ratios of code blocks that are identical to those provided by the OpenZeppelin project.

Conclusion: Due to the immutability of smart contracts, as well as the impossibility of reverting transactions once they are deemed final, we conclude that the aforementioned findings yield implications to the security, development, and usage of smart contracts.

1.2.4 A First Look at the Integration of Machine Learning Models in Complex Autonomous Driving Systems(EASE)

BackGround: Autonomous Driving System (ADS) is one of the most promising large–scale machine learning powered systems. Hence, ADS has attracted much attention from academia and practitioners in recent years. Despite extensive study on ML models, it still lacks a comprehensive empirical study towards understanding the ML model roles, peculiar architecture, and complexity of ADS.

Aim: In this paper, researchers conduct an in–depth case study on Apollo, which is one of the state–of–the–art ADS, widely adopted by major automakers worldwide to reveal the integration of the underlying ML models and code logic in Apollo.

Method: The researchers study the Apollo source code and present the underlying ML model system architecture and inspect Apollo in a dynamic view to evaluate the operating status.

Result: The researchers present their findings on how the ML models interact with each other, and how the ML models are integrated with code logic to form a complex system. Besides, they notice the heavy use of model–relevant components and the lack of adequate tests in general.

Conclusion: The study reveals potential maintenance challenges of complex ML–powered systems and identifies future directions to improve the quality assurance of ADS and general ML systems.

1.2.5 Mining Questions Asked about Continuous Software Engineering: A Case Study of Stack Overflow（EASE）

BackGround: With the growing popularity of rapid software delivery and deployment, the methods, practices and technologies of Continuous Software Engineering (CSE) are evolving steadily. This creates

the need for understanding the recent trends of the technologies, practitioners'challenges and views in this domain.

Aim: In this paper, we present an empirical study aimed at exploring CSE from the practitioners' perspective by mining discussions from Q&A websites.

Method: We have analyzed 12,989 questions and answers posted on Stack Overflow. Topic modelling is conducted to derive the dominant topics in this domain. Further, a qualitative analysis was conducted to identify the key challenges discussed.

Result: We identified 32 topics of discussions, among which "Error messages in Continuous Integration/Deployment" and "Continuous Integration concepts" are the most dominant. We also present the most challenging areas in this domain from the practitioners'perspectives.

Conclusion: Whilst the trend of posted questions is sharply increasing, the questions are becoming more specific to technologies and more difficult to attract answers.

1.3 Reasons

The case study we selected satisfies the following three criteria:

1.In the selected articles, case studies are the main research method. The introduction of the method accounts for a relatively high proportion of the overall length of the article.

2.The selected article needs to have a clear description of the key elements and main process of the case study.

3.The selected case studies need to draw one or more specific and clear conclusions. Every conclusion should be supported by research findings.

1.4 Problems and gained experiences

1.4.1 Problems

1. In both abstract and the main body of some reports, some key elements are missing. Meanwhile, some of key elements are not specified clearly which make it a tough task to sum them up.

2. Incomplete correspondence between problems and cases. For instance, in the paper describing Apollo, the major tasks are divided into two parts, basing on static static perspective and dynamic perspective respectively. However, dynamic perspective is not reflected by RQs, but becomes another section alone.
3. In some reports, research methods are too complex and diverse to summarize cushly.

1.4.2 Solutions

1. If there are relevant descriptions in the report, the elements shall be sorted out and extracted after reading the report. If there is no relevant description in the article, the element is discarded.
2. Reading the specific contents of the disputed cases in detail and re classify the case types helps. After further exploration, we consider that the chief methodology of the tasks related to dynamic perspective is experiment instead of case study.
3. We sort out the common methods of case studies, summarize their key characteristics, and match them with the methods described in the article.

2.Study Analysis and Comparison

2.1 Study Analysis

CHARACTERISTICS	Automated issue assignment: results and insights from an industrial case (EMSE)
Objective	In this work, we automate the process of issue assignment by using data mining approaches and share our experience gained by deploying the resulting system, called IssueTAG, at Softtech.
Participants	Softtech is the largest software company of Turkey owned by domestic capital. Being an ISO–9001–certified subsidiary of the largest private bank in Turkey, called IsBank,Footnote Softtech receives an average of 350 issue reports every day from the field. IssueTAG has been making all the issue assignments since its deployment on Jan 12, 2018.
Case Amount	Single
Organization	Embedded
Triangulation	Theory triangulation

Data collection techniques	direct methods
Source	physical artifacts + participant–observation
Interview	/
Interview mode	/
Research question	RQ1: How does automated issue assignment compare to manual issue assignment in practice? RQ2: Can the issue assignments made by the underlying data mining model be explained in a non–technical manner? RQ3: Can the deteriorations in the assignment accuracies be automatically detected in an online manner? RQ4: Is IssueTAG perceived as useful by the end–users?
Qualitative Analysis	RQ1: We first observed that after IssueTAG was deployed, the daily assignment accuracies dropped slightly. We, however, observed that the accuracy of an automated issue assignment system does not have to be higher than that of manual assignments in order for the system to be useful. RQ2: Based on the explanations created for the assignments, only one of the assignments was found “untrustworthy.” And, this assignment was, indeed, an incorrect assignment made by IssueTAG. RQ3: We observed that as the deterioration amount increased, the detection time tended to decrease, i.e., the proposed approach tended to detect the deteriorations faster. RQ4: The results of the survey strongly suggest that IssueTAG meets its business needs with high quality.

Validities	Automated issue assignment: results and insights from an industrial case (EMSE)
Conclusion Validity	All the issue reports we used in the experiments were the real issue reports collected from the field. After the deployment of IssueTAG, once an issue report was created by an IT–HD clerk, the assignment was automatically made by the system.
Construct Validity	To circumvent the construct threats, we used the well–known accuracy metric throughout the paper to evaluate the quality of the issue assignments. We have also complemented the accuracy results with other well–known metrics, namely precision, recall, and F–measure, as we see fit.
Internal Validity	To circumvent the internal threats that may be caused by implementation errors, we used well–known and frequently used tools. In particular, we used the Python scikit–learn tool for preprocessing the issue reports and extracting the features; the scikit–learn and mlxtend tools for training the classification models; the lime tool for creating the LIME–based explanations for the assignments; and the ruptures tool for PELT–based change point detection.
External Validity	One external threat is that IssueTAG was deployed at Softtech/IsBank only. Another possible threat is that issue reports at IsBank (thus, the ones used in this work) are created by the IT–HD clerks.

CHARACTERISTICS	On the impact of using trivial packages: an empirical case study on npm and PyPI (EMSE)
Objective	This paper has extended our previous work to strengthen the empirical evidence on the use of trivial packages by replicating and extending our study on the Python Package Index.
Participants	Our study involves more than 500,000 npm packages and 38,000 JavaScript applications and 63,000 PyPI packages and 14,000 Python applications. The study also contains survey results from 125 JavaScript and Python developers.
Case Amount	Multiple
Organization	Embedded
Triangulation	Data (source) triangulation
Data collection techniques	direct methods
Source	physical artifacts + participant–observation
Interview	/
Interview mode	/
Research question	RQ1: How Many of npm's & PyPI's Packages are Trivial? RQ2: How Many Applications Depend on Trivial Packages?
Qualitative Analysis	RQ1: Out of the 501,001 npm packages we mined, 80,232 (16.0%) packages are trivial packages. Our analysis reveals that out of the 63,912 *PyPI* packages we analyzed, 6,759 (10.6%) packages are trivial packages in the *PyPI* package management platform. RQ2: We find that of the 38,807 applications in our dataset, 10,139 (26.1%) directly depend on at least one trivial package.Our analysis showed that out of the 14,717 examined Python applications, 1,024 (6.9%) were found to depend on one or more trivial *PyPI* package.

Validities	On the impact of using trivial packages: an empirical case study on npm and PyPI (EMSE)
Construct Validity	To determine what is considered to be a trivial package, we conducted an experiment with JavaScript and Python developers who are mostly students (undergraduate and graduate students) with some professional experience. While this may not present professional developers per se (Sjoberg et al. 2002), prior work has shown that experiment with students will provide the same results as professional developers in software engineering domain.
Internal Validity	Internal validity concerns factors that may have influenced our results such as our datasets collection process. To study the reasons for and drawback of using trivial packages, we surveyed developers. There is potential that our survey questions may have influenced the replies from the respondents. However, to minimize such influence, we made sure to ask for free–form responses and we publicly share our survey and all of our anonymized survey responses.
External Validity	External validity considers the generalization of our findings. All of our findings were derived from open source JavaScript applications and *npm* packages and its replication on Python and *PyPI* packages. Even though we believe that the two studied package management platforms are amongst the most commonly used ones, our findings may not generalize to other platforms or ecosystems. That said, historical evidence shows that examples of individual cases contributed significantly in areas such as physics, economics, social sciences and even software engineering. We believe that strong empirical evidence is built from both studies on individual cases and studies on large samples.

CHARACTERISTICS	Code cloning in smart contracts: a case study on verified contracts from the Ethereum blockchain platform(EMSE)
Objective	Our goal is to quantify the amount of clones in Ethereum, understand key characteristics of clone clusters, and determine whether smart contracts contain pieces of code that are identical to those published by OpenZeppelin.
Participants	Ethereum is a blockchain platform. A blockchain platform is a distributed, chronological database of transactions that is shared and maintained across nodes that participate in a peer–to–peer network.
Case Amount	Single
Organization	Embedded
Triangulation	Data (source) triangulation
Data collection techniques	direct methods+independent analysis
Source	archival records
Interview	/
Interview mode	/
Research question	RQ1: How frequently are verified contracts cloned? RQ2: What are the characteristics of clusters of similar verified contracts? RQ3: How frequently code blocks of verified contracts are identical to those from OpenZeppelin?
Analysis	RQ1: 79.2% of the verified contracts are clones. There is a large number of clone clusters. Nevertheless, a small portion of clusters encompasses the vast majority of contracts. RQ2: 9 out of the top–10 largest clusters are token managers. 1 of the top–10 largest clusters is a Token Locker. RQ3: 36.3% of the verified contracts have at least one code block in their code file that is identical to an OpenZeppelin code block. 26.3% of all 165,005 code blocks extracted from verified contracts are identical to OpenZeppelin code blocks.

Validities	Code cloning in smart contracts: a case study on verified contracts from the Ethereum blockchain platform(EMSE)
Construct Validity	We detect clones using Deckard. Instead of arbitrarily defining configuration parameters, we performed a careful sensitivity analysis (Section 5). Our sensitivity analysis relied on 10 known clones (control group). Choosing a different control group would possibly lead to the selection of a different *similarity* threshold, which would in turn influence the set of clone clusters detected by the tool. Using other parameter configurations, as well as other clone detectors, is a future work endeavor.
External Validity	The population investigated in this study consisted of all verified smart contracts available on Etherscan at the time of data collection. Therefore, our results may not generalize to all smart contracts in Ethereum. However, the goal of this paper is not to build a theory that applies to all contracts, but rather to make developers and researchers aware that cloning is the *modus–operandi* of developing smart contracts. Nevertheless, additional replication studies are required in order to generalize our results to non–verified contracts as well as smart contracts deployed in other blockchain platforms, such as EOS and POA.

CHARACTERISTICS	A first look at the integration of machine learning models in complex autonomous driving systems(EASE)
Objective	In this paper, the target is to conduct an in–depth case study on Apollo 5.0, including the ML models' interactions with the system, relationship between the ML models and code logic at the integration level, Apollo's ML model usage and the current testing effort.
Participants	In this paper,researchers conduct an in–depth case study on Apollo 5.0, which is one of the most advanced ADS systems in the world, along with the 28 ML models contained by Apollo. Moreover, the developpers of Apollo also helps to deploy test execution and confirm the validity of result.
Case Amount	Single
Organization	Embedded
Triangulation	Data (source) triangulation + Methodological triangulation
Data collection techniques	Direct methods + independent analysis

Source	Documentation + archival records + participant–observation + source code + ML model
Interview	\
Interview mode	\
Research question	RQ1: What is the ML model architecture and relevant information flow in Apollo? RQ2: What are the relationship and interaction between code modules and ML models?
Qualitative Analysis	RQ1: Typical ML models, especially deep neural networks, are widely used in Apollo. 28 ML models were found utilized by Apollo. The ML models can be categorized into four major categories: (1) traffic light perception, (2) lane perception, (3)obstacle detection, and (4) trajectory prediction. For each model category, Apollo often provides several ML models. RQ2: Researchers summarize the common ML model interactions in Apollo as below: (1) All the output of one ML model is used exclusively and entirely by another ML model as input. (2) The output of one ML model is used to post–process the output of another ML model. (3) The outputs of several ML models are combined together as another ML model's input.

Validities	A first look at the integration of machine learning models in complex autonomous driving systems(EASE)
External Validity	Currently, Apollo has adopted by major automakers. However, the results may still not generalize to other ADS or ML systems. Their findings provide an initial overview on the architecture of ML systems and how ML models are integrated with code logic. Since ML systems are becoming more critical but common knowledge about development remains vague, future studies are required to investigate the design and challenges of other ML systems.
Internal Validity	Much of analysis relies heavily on manual inspection of the Apollo source code and documentation. Although many of their results are confirmed by Apollo developers, there may still be biases in their results. To mitigate the issue, the third and fourth authors also verify the results of the manual inspection.

CHARACTERISTICS	Mining Questions Asked about Continuous Software Engineering: A Case Study of Stack Overflow(EASE)
Objective	Our study aims at filling this important gap in the literature for contributing to the field by investigating the trend of the posted questions, the dominant topics of the discussions and the challenges reported on SoF as Q&A.
Participants	We extracted the data from SoF using CSE related tags on SoF. Looking into the list of the available tags, we found 8 tags as highly relevant to the context of our study (e.g., shown in Table 1). These tags were chosen based on the continuous practices of CSE that are of a technical nature and relevant popular DevOps tags. We intentionally did not include the technology specific tags (e.g., 'Jenkins', 'docker') to avoid biasing the data towards specific tools. Using 'Stack Exchange API' we retrieved the data of 14,251 questions with their answers. The data was extracted on the 5th of September 2019. We further explored the dataset and removed duplicated records that were created due to having multiple tags. Hence, the number of questions were reduced to 12,989.
Case Amount	Single
Organization	Embedded
Triangulation	Methodological triangulation
Data collection techniques	independent analysis
Source	Documentation
Interview	\
Interview mode	\
Research question	RQ1: How successfully are the questions related to CSE answered in SoF? RQ2: What are the dominant, popular, difficult and easy topics of discussion on CSE in SoF? RQ3:What are the main themes of CSE challenges in SoF?
Qualitative Analysis	The trend of unsuccessful questions, however, demonstrates a drastic increase since 2014. From almost the end of 2017 the trend of unsuccessful questions has preceded successful and ordinary questions.

Validities	Mining Questions Asked about Continuous Software Engineering: A Case Study of Stack Overflow(EASE)
External Validity	/
Internal Validity	We realize that our results could have limitations due to our data collection approach. We collected the data of the Q&A about continuous software engineering from SoF using the assigned tags. Whilst this approach is effective to find the relevant posts and decrease the chance of including false positives, it can create limitations in comprehensiveness of data. Using tag–based approach requires all the potentially related tags have been considered.

2.2 Study Comparison

1. None of the five articles contained an interview section. It can be seen that the interview part is an unnecessary part of the case study. When the research object is human, the interview is more meaningful.
2. The five articles are very diverse, ranging from studies on people to articles on experimental code. In addition, there are articles on joint research on various objects such as documents and experimental code.
3. Methodological triangulation is the most widely used triangulation. It can be seen that the use of multiple methods to describe the same object can lead to higher quality results.
4. Embedded organizations are the most widely used organizations. In fact, the single organization was not used much in case studies. A mix of research methods is more helpful in answering different research questions.
5. When analyzing the code, there is often an external validity threat, that is, the conclusion obtained by observing the research on a certain piece of code or a certain system, due to the incompleteness of the observation object, its representativeness and generalizability are limited. This problem can be partially resolved by consulting an expert.
6. Code, interviews, and documentation are the most common sources of data. Most articles have more than one source of data.
7. Combining quantitative and qualitative analysis is an important feature of a good case study. This feature can be reflected in the methodological triangulation.
8. independent analysis and direct interview are the two most common methods of data collection.
9. Among the research questions of case study, there are generally considerations such as "To what extent". Carrying out quantitative research around such research questions is an important part of case study.