DevOps and ci/cd Pipeline: exploring the edge computing landscape

Doctoral Defense Preview

Igor Muzetti Pereira

Advisors: Tiago Carneiro
Eduardo Figueiredo

June 21, 2024

Table of Contents

Introduction

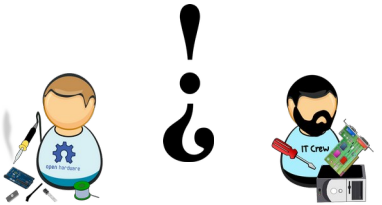
Contextualization

- Organizations and communities developing IoT systems rely on **software components** as one of the **main assets** in their products
- When the systems become more complex, these entities **need to adapt** [Luz et al., 2019]
- These entities have teams that develop **hardware** that traditionally uses rigorous and plan-oriented methodologies [Lindgren and Münch, 2016]
- Moreover, they have software development and operations teams that are more familiar with to agile methodologies

Contextualization

- On one hand, an IoT system project requires **ongoing cooperation between development, operations, and hardware teams**
- On the other hand, there may be **resistance in hardware teams** to using **agile practices** in their activities
- IoT systems are composed of **addressable objects (things)** that **interact with each other and with users**, **detecting and processing data** to achieve specific goals [Jacobson et al., 2017]
- These systems have **different embedded components**, leading to the creation and use of architectural patterns for various applications [Wu et al., 2010]

Problem



- IoT systems have **several components** and require the **integration of multiple experts** and **development approaches**
- Software engineering **must adapt to ensure the quality** of IoT systems
- **Component variability** makes choosing technologies and practices for IoT embedded projects **complex**
- The specific **challenges of adapting the CI/CD pipeline** for IoT embedded systems are not understood

Motivation

- The CI/CD pipeline **can expedite build, test, and delivery,** improving quality and time to release
- **Efficient organizational structures** are necessary to improve **communication and delivery speed** in IoT embedded systems
- The **integration of hardware, software, and infrastructure** is not just a component but a critical pillar of **success in IoT systems**
- Understanding and overcoming the **challenges of CI/CD in IoT embedded systems** can provide valuable insights for **practitioners and researchers**, advancing software engineering in this domain

General Goal

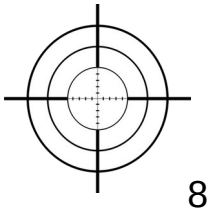
Investigate the challenges of software development, maintenance and continuous delivery

for the purpose to clarify to software engineering community

with respect to proposing approaches to overcome these challenges

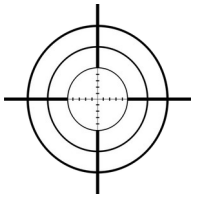
from the point of view of researchers, professionals and practitioners

in the context of IoT embedded systems.



Specifics Goals

- Identify DevOps practices and tools used in IoT embedded projects reported in the literature and by professionals
- Elucidate teams' benefits and challenges when adopting DevOps in IoT embedded projects
- Select a subset of practices and tools that promote the quality of FLOSS (Free/Libre Open-Source Software) projects from these domains during DevOps adoption
- Verify developers performance when using these practices and tools on a project



Theoretical Reference

Background

- **DevOps** is a collaborative and multidisciplinary organizational effort to automate the continuous delivery of new software updates while guaranteeing their correctness and reliability [Leite et al. 2019]
- The **CI/CD pipeline** is an automated sequence of actions that involves source code from the machine of the developer to the target device that is in production and available to system users
- **IoT embedded systems** are hardware and software components that execute continuously and react to events in the environment

Background

- We are considering an **architecture for IoT systems** with **4 layers**. Each layer refers to different levels and plays a specific role in data processing and management [Zhang et al., 2018] [Kumar and Agrawal, 2023]
  - The **sensors and devices layer** is closest to the physical environment
  - The **edge layer** processing data close to the source location, reducing latency, and relieving the load on the network
  - The **fog layer** provides more advanced computing, storage, and communication resources, typically on local servers
  - The **cloud layer** handles intensive processing and long-term storage and provides advanced services like big data analytics, machine learning, and web services

Background

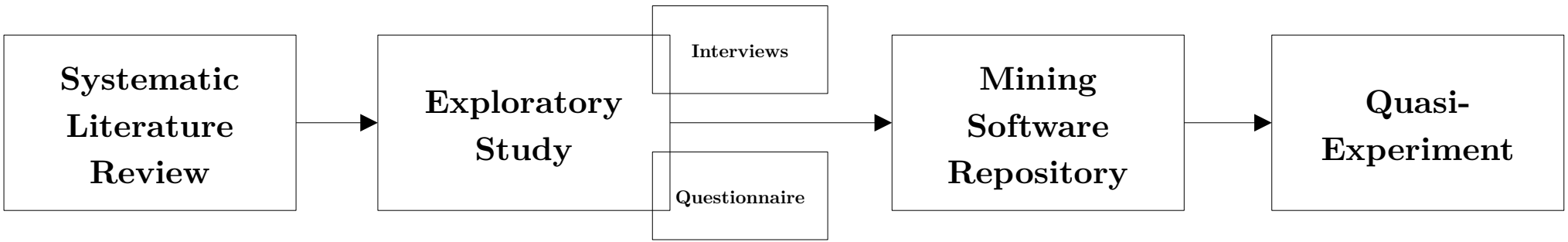
- **GitHub** is a source code hosting platform for version control and collaboration. It allows developers to work together on projects in a geographically distributed manner
- **GitHub Actions** enables task automation involving several components, such as PR, commits, and issues, that are applied to facilitate the automation of builds, tests, deliveries, deploys, and communication

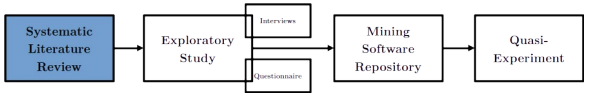
Related Works

- CI/CD in different contexts, such as cloud, IoT, CPS, and edge and fog computing
- These studies reveal challenges, practices, benefits, and trends in applying CI/CD in complex software development environments
- Some key points include heterogeneity in continuous delivery workflows, automation, and tools, testing in simulated versus real environments, hardware and software management, balancing regulatory compliance and agile development, and flexible and interdisciplinary approaches

Experimental Procedures

Steps of research experimental procedures

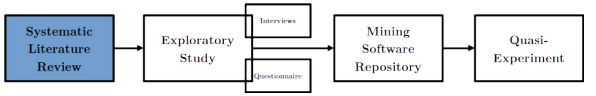




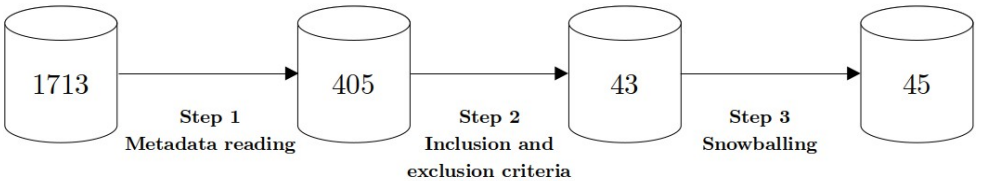
Search of the SRL

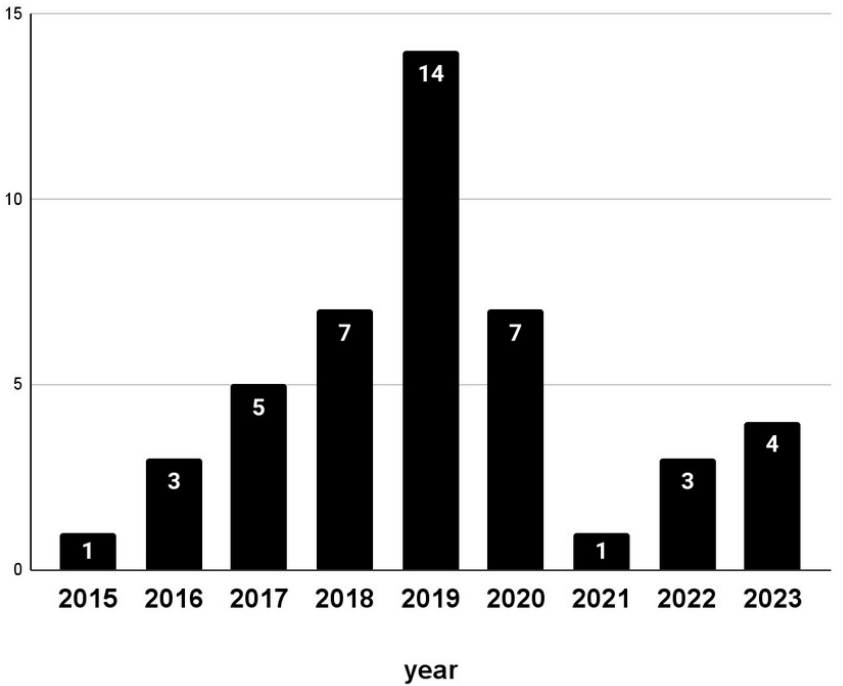
Data Source	Address	Results
IEEE Xplore	https://ieeexplore.ieee.org/	120
ACM Digital Library	https://dl.acm.org/	402
Science Direct	https://www.sciencedirect.com/	360
Springer	https://link.springer.com/	567
Wiley	https://onlinelibrary.wiley.com/	264
Total		**1713**

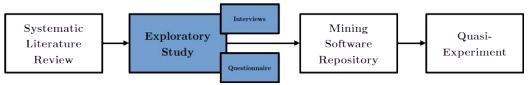
(("internet of things" OR iot) AND ("devops"))



Search Results

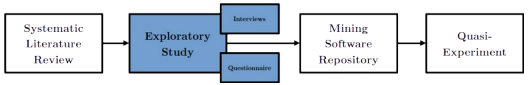






Exploratory Study Design

Interview	Questionnaire
91 professional invited	60 professional invited
31 responded	30 responded
Free and informed consent forms	First section about participant's backgroud
Encouraged to speak freely	Second section with close questions
Pilot study	
Google Meet and Audacity	
Lasted between 30-50 minutes	
Each interview was transcribed	
Open coding for analysis	

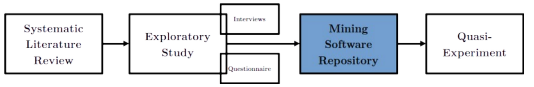


Background of Participants

Experience	Interview	Questionnaire
Less than one year	0%	3.4%
Between one and three years	0%	10.3%
Between three and five years	32.2%	20.7%
More than five years	67.8%	65.5%
Area		
Development	25.8%	33.3%
Operation	48.4%	36.7%
Hardware	25.8%	30.0%

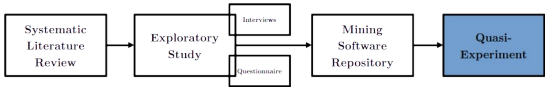
most are experienced

balance among the areas

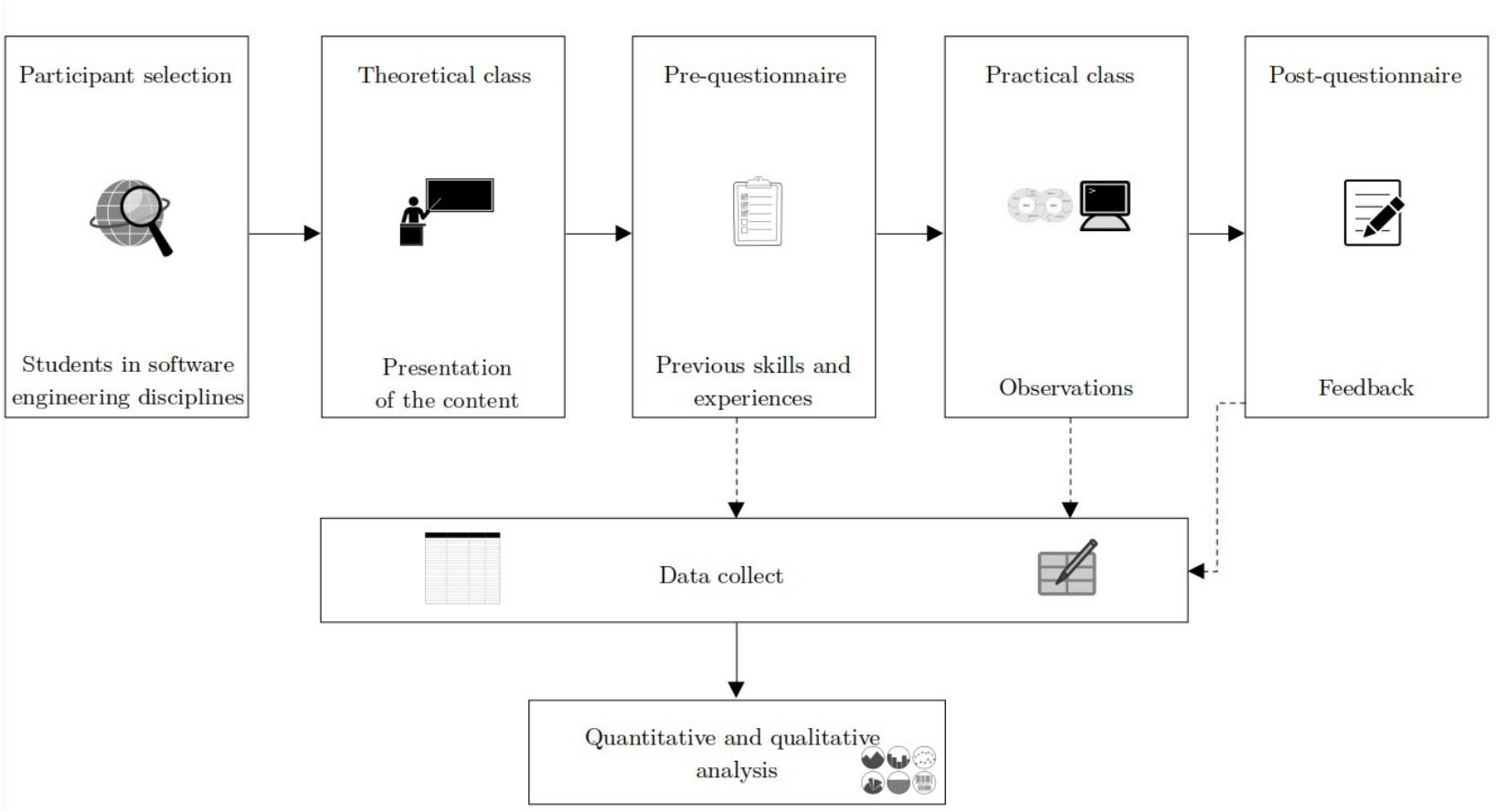


Mining Software Repository (MSR)

- We selected and quantitatively analyzed:
  - 11 metrics from repositories of 25 IoT-related projects
    - Popularity: stars, forks, issues and watchers
    - Delivery: release, commits, branches, closed and merged PR, closed but not merged PR, and open PR
    - Community: contributors, truck factor, and one-time contributor
- Used the K-Means to clustering and understanding the characteristics of the repositories
- Explored evidence of use in GHA workflows (YAML files)
- Applied open coding to issues to understand how these communities discuss the CI/CD pipeline

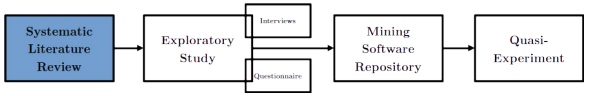


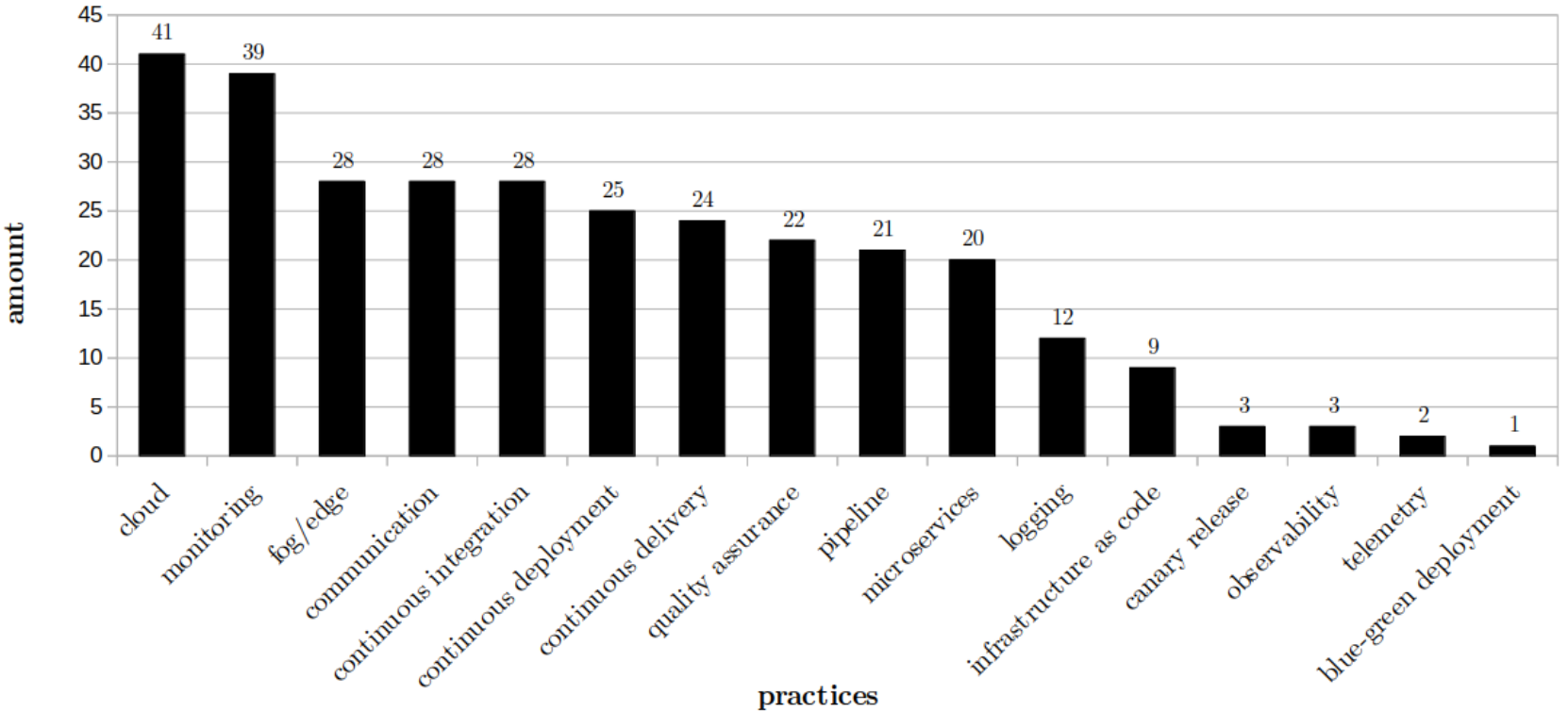
Quasi-Experiment



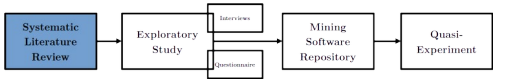
Results

RQ1. What are the DevOps topics investigated in the context of IoT software systems?

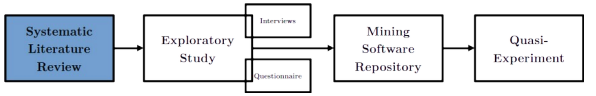




plan	communication
develop	architecture languages and frameworks ide library and toolkit storage protocols and communications formats operation system hardware servers network build and interpreter testing version control container continuous integration
deliver	continuous delivery / deployment infrastructure as code source code hosting
operate	measurement processing and analysis

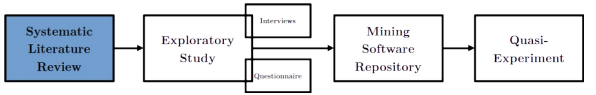


RQ2. What are the benefits of adopting DevOps in IoT software systems?

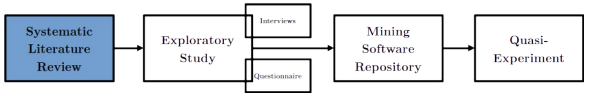


Benefit	Category
B01	DevOps framework proposals for IoT
B02	Use of cloud and fog computing
B03	Economics with the use of containers
B04	Microservices induce the formation of small teams
B05	Reference architectures DevOps to IoT systems

RQ3. What the reported challenges of adopting DevOps in IoT systems?



Challenge	Category
C01	Manage quality of service
C02	Select devices
C03	Continuous feedback between teams
C04	Apply a deployment pipeline
C05	Continuous documentation
C06	Need for e-shaped professionals



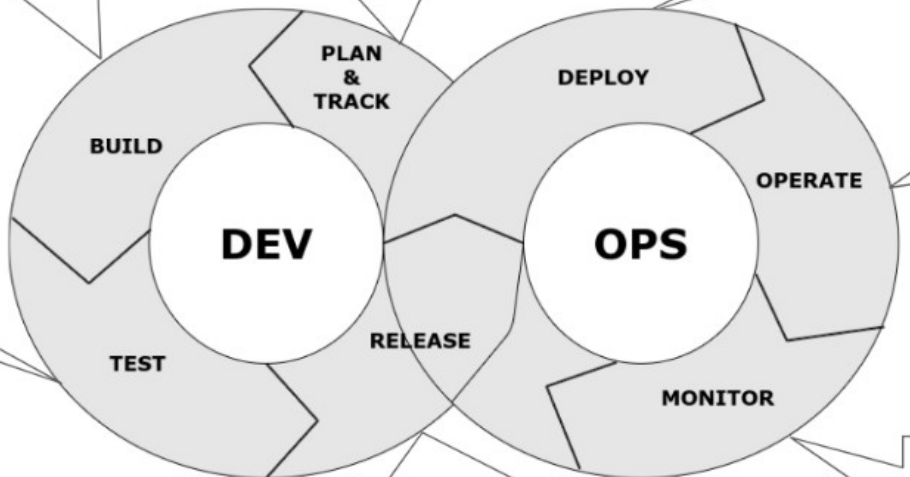
2. Frontend and backend development in client and server applications. Configuration of software embedded. Hardware construction and physical support.

1. Design and specification of software, hardware, and physical requirements. Selection of libraries, languages, frameworks, and protocols.

5. Configuration and deployment management in a production environment.

3. Unit, static, and security tests in software. Tests on connectivity devices, components, and hardware cards.

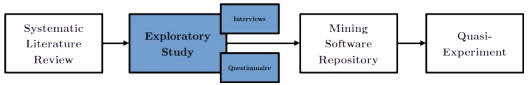
6. Provisioning and orchestration.



4. Development and operations teams integrate their work so that problems are detected and corrected. Performance, acceptance, and usability tests on IoT prototypes.

7. Aggregate information about the stages of the value flow. Act according to the analysis of the results.

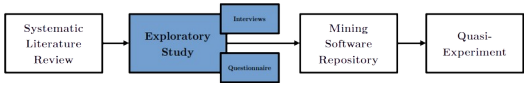
RQ1. What are the differences in developing embedded systems with a traditional process compared to a process that seeks continuous delivery and develops IoT systems?

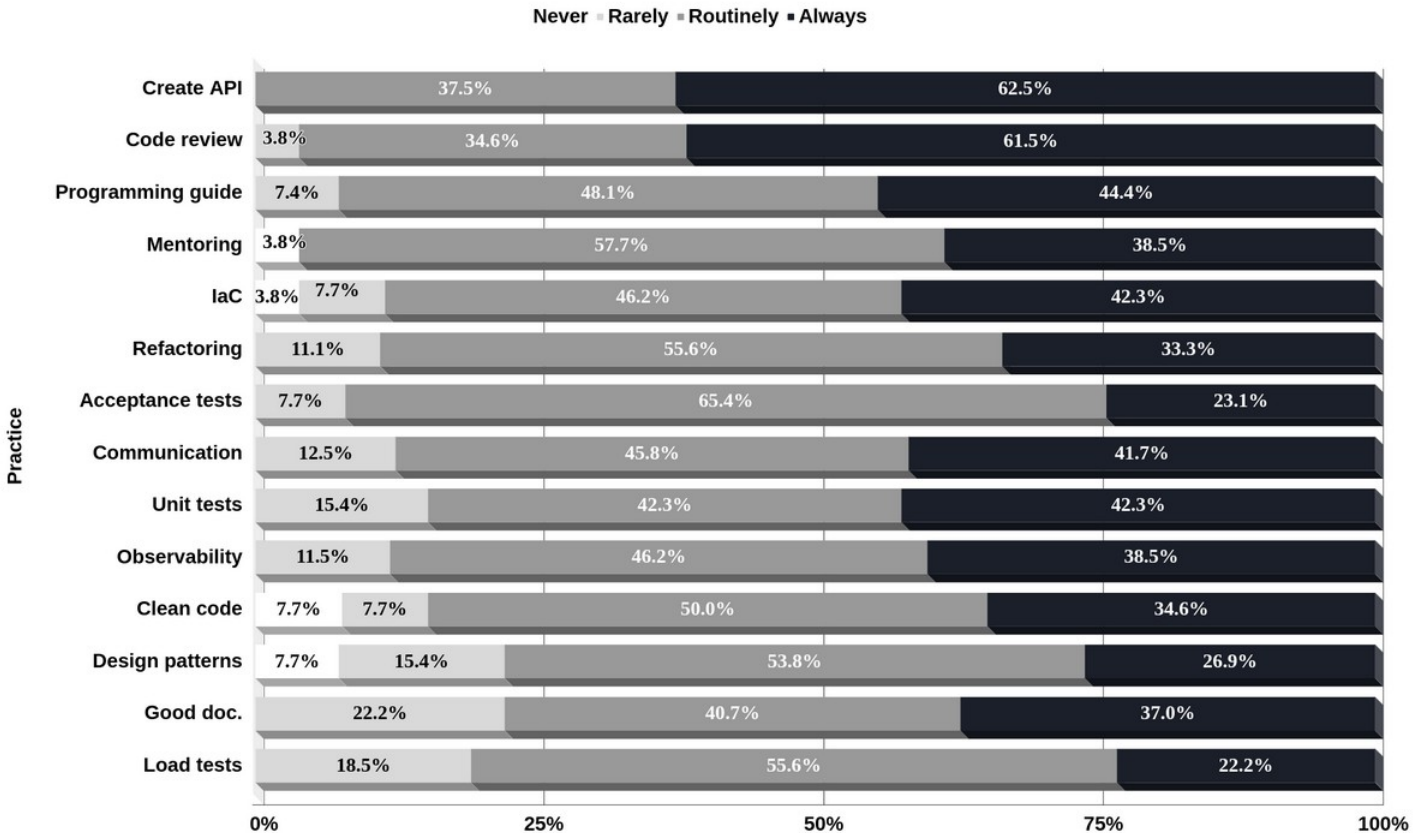


- Some companies develop their prototypes; others even import its
- The final hardware is **mainly** purchased from China
- **Every IoT solution is different** from another
- Use of microservices
- Use of IaC to obtain stable environments
- Different delivery/deployment strategies

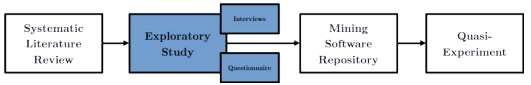
answers more related to technical and business aspects

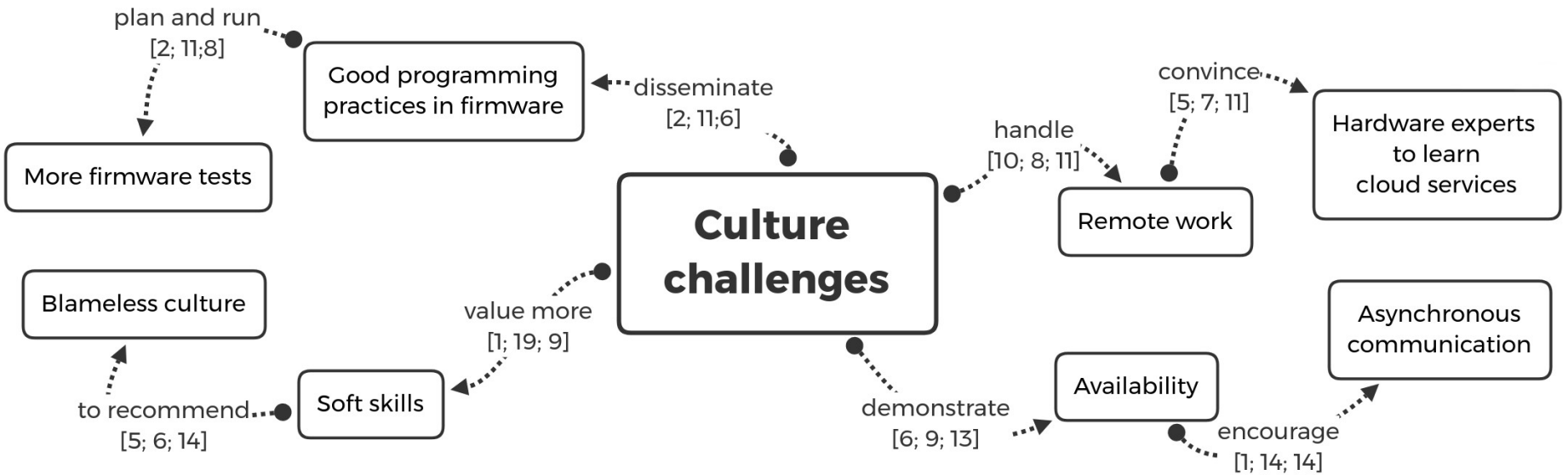
RQ2. Which quality practices are most used in the pursuit of continuous delivery in developing IoT systems?



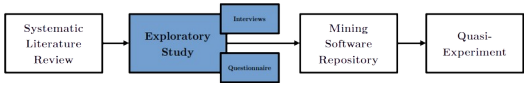


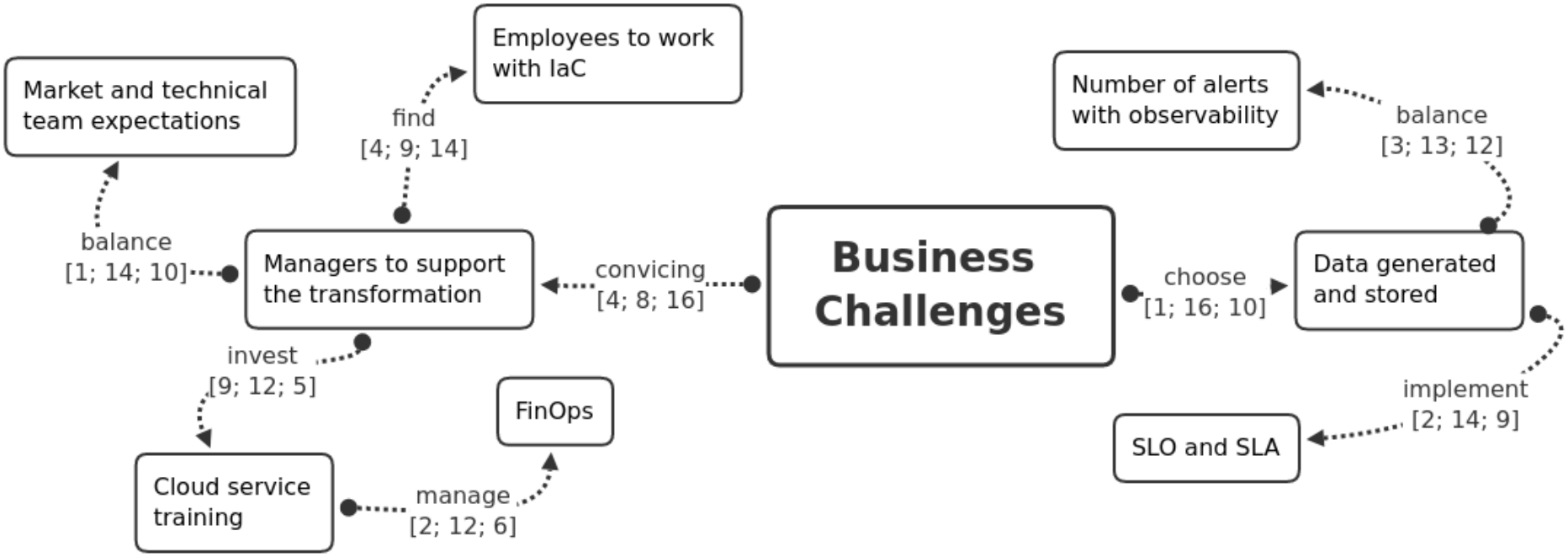
RQ3. How are perceived challenges faced in the quest for continuous delivery?



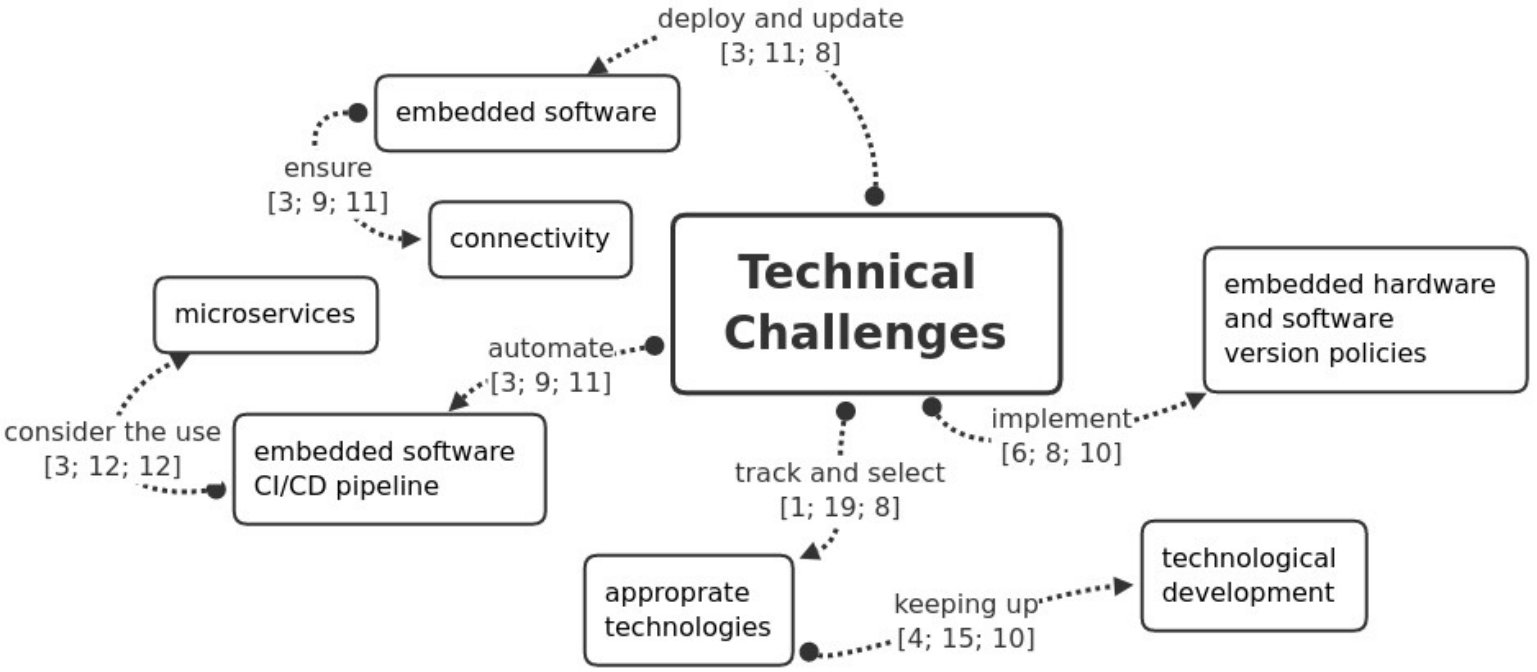


RQ3. How are perceived challenges faced in the quest for continuous delivery?

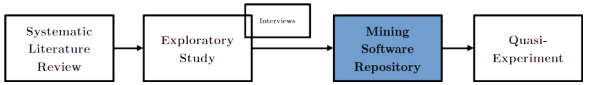


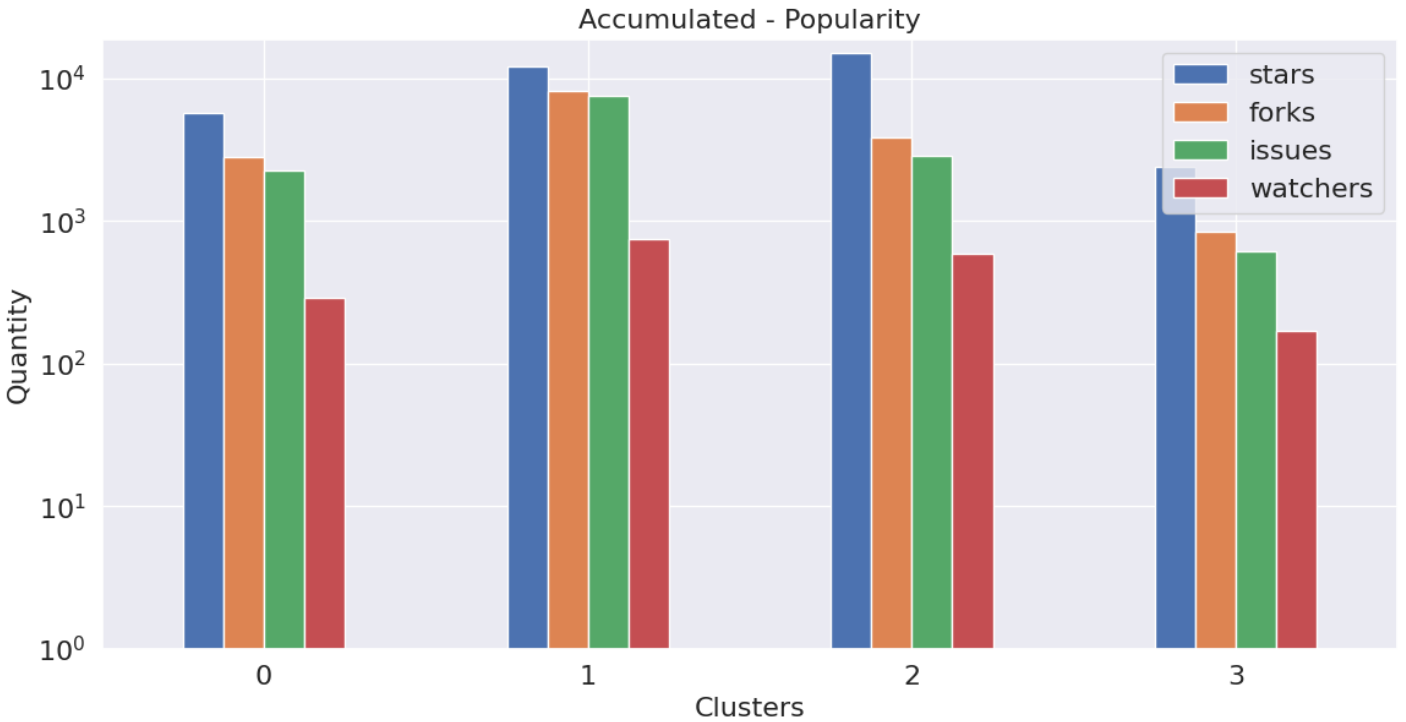


RQ3. How are perceived challenges faced in the quest for continuous delivery?

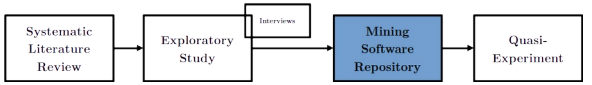


RQ1. What are the characteristics of FLOSS IoT-related projects?



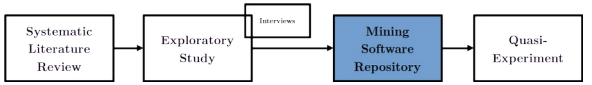


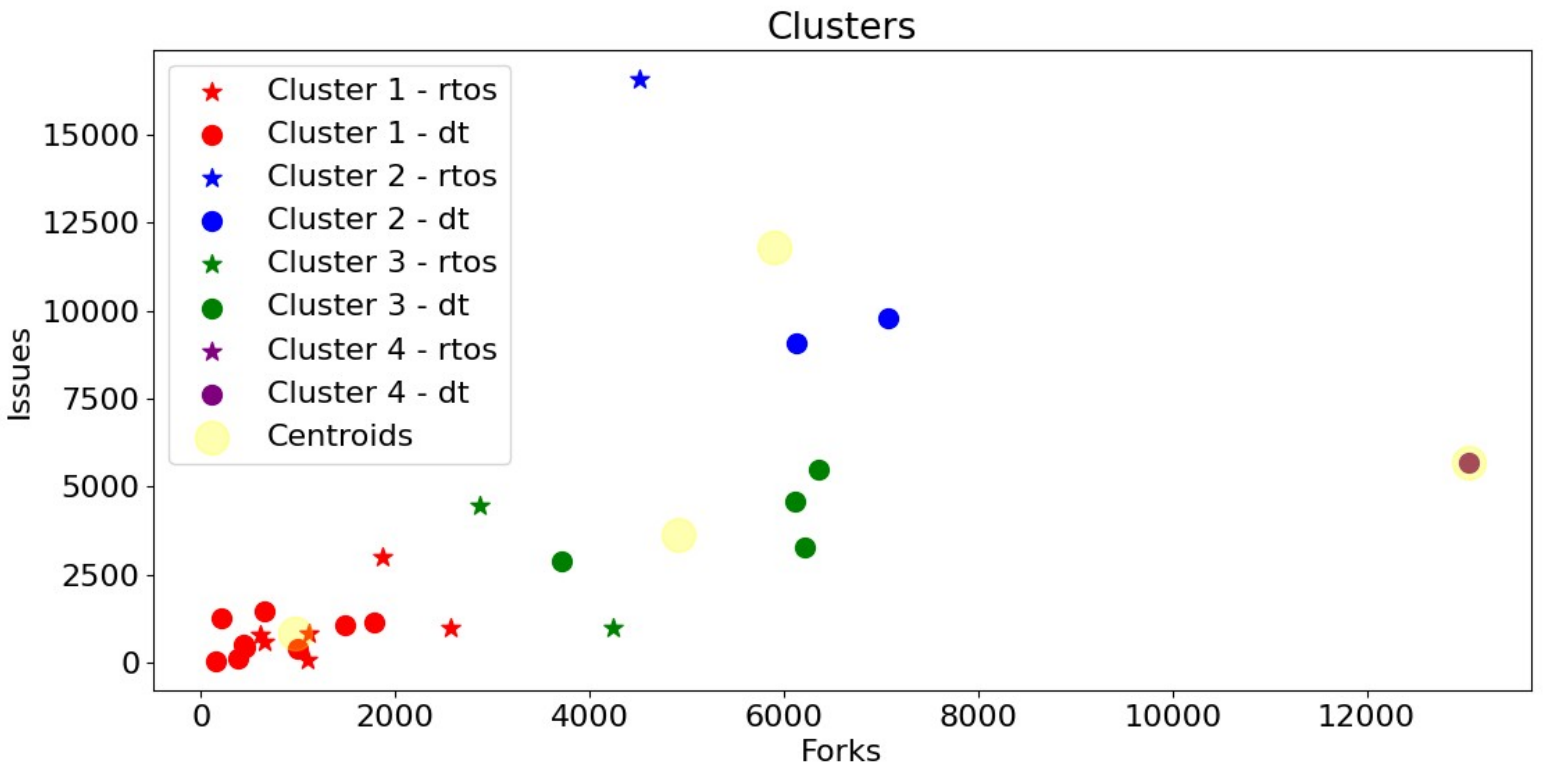
RQ1. What are the characteristics of FLOSS IoT-related projects?



Project	Kind	General	Popularity	Delivery	Community
zephyrproject-rtos/zephyr	RTOS	2	2	1	2
ARMmbed/mbed-OS	RTOS	3	1	2	1
RIOT-OS/RIOT	RTOS	3	1	2	3
contiki-os/contiki	RTOS	0	1	3	0
contiki-ng/contiki-ng	RTOS	0	1	3	3
RT-Thread/rt-thread	RTOS	3	1	0	1
aws/amazon-freertos	RTOS	0	1	0	0
apache/nuttx	RTOS	3	1	2	3
FreeRTOS/FreeRTOS	RTOS	0	1	3	0
esp8266/Arduino	Library(DT)	1	0	0	1
arduino/Arduino	IDE (DT)	1	0	3	3
espressif/arduino-esp32	Library (DT)	1	0	3	1
arduino/arduino-ide	IDE(DT)	0	1	3	0
espressif/ESP8266_RTOS_SDK	Framework(DT)	0	1	3	0
espressif/esp-idf	Framework(DT)	1	0	3	1
jerryscript-project/iotjs	Framework(DT)	0	1	3	0
jerryscript-project/jerryscript	Library(DT)	0	1	0	0
micropython/micropython	Language(DT)	1	3	0	1
hybridgroup/gobot	Framework(DT)	0	1	3	0
hybridgroup/cylon	Framework(DT)	0	1	3	0
hybridgroup/artoo	Framework(DT)	0	1	3	0
Tencent/ncnn	Framework(DT)	1	3	3	3
rwaldron/johnny-five	Framework(DT)	1	3	3	0
cesanta/mongoose-os	Framework(DT)	0	1	3	0
adafruit/circuitpython	Language(DT)	3	1	0	0

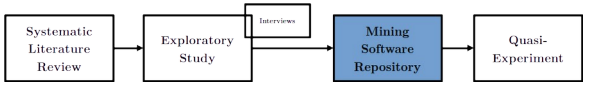
RQ1. What are the characteristics of FLOSS IoT-related projects?

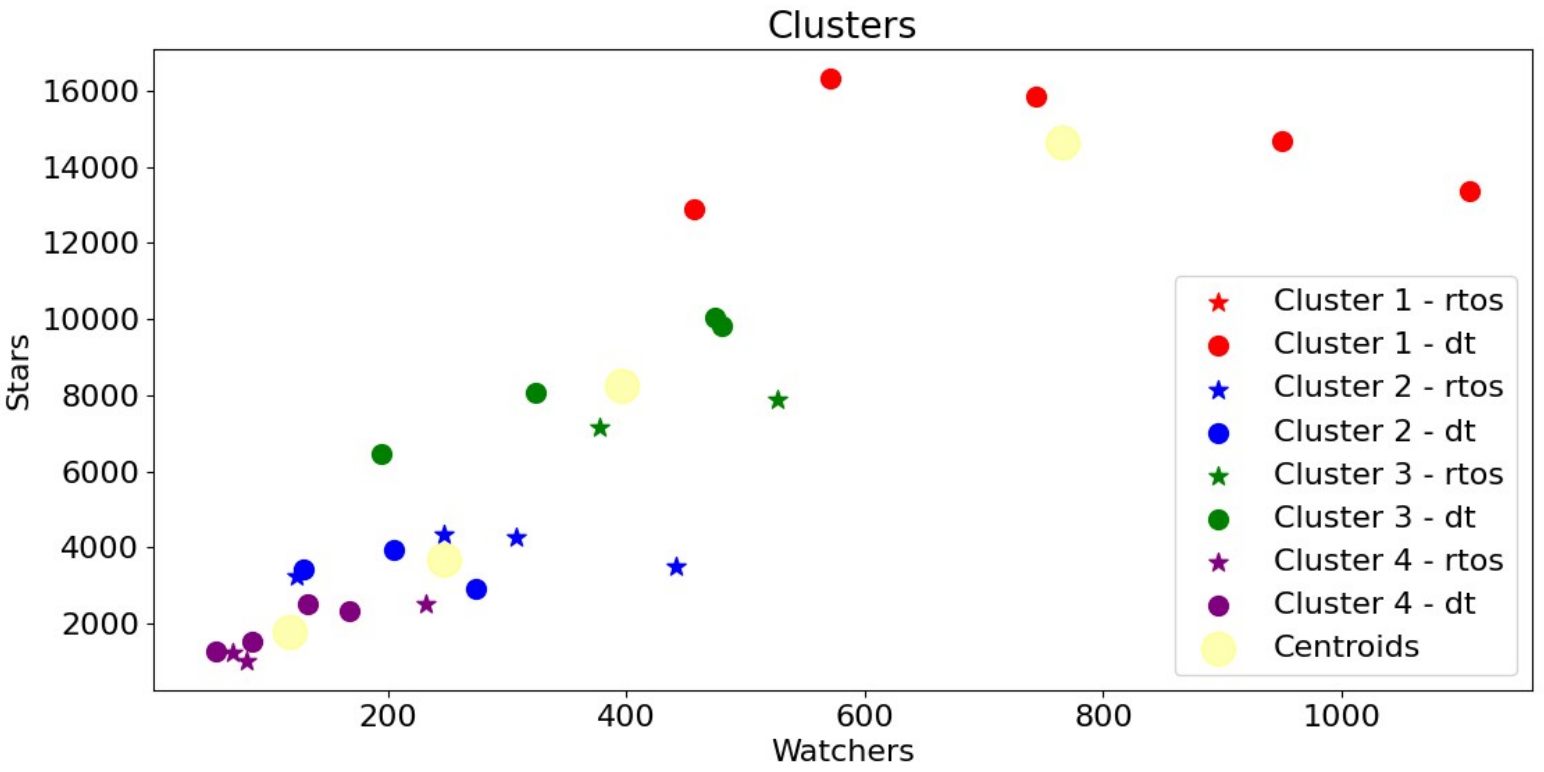




Spearman: 0.825
Silhouette: 0.57

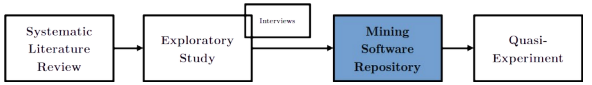
RQ1. What are the characteristics of FLOSS IoT-related projects?





Spearman: 0.909
Silhouette: 0.59

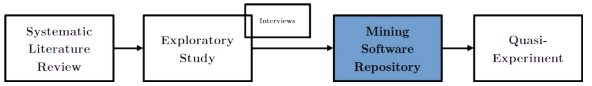
RQ2. How do FLOSS IoT-related projects structure their CI/CD pipeline?



Event	%Workflows	%Repositories
push	74.63%	72.00%
pull_request	71.98%	72.00%
release	25.82%	44.00%
schedule	15.93%	36.00%
workflow_dispatch	10.99%	32.00%
issues	9.34%	28.00%
pull_request_target	4.40%	16.00%
workflow_run	2.20%	12.00%
repository_dispatch	1.65%	4.00%
workflow_call	1.65%	4.00%
issue_comment	1.10%	8.00%
pull_request_review	0.55%	4.00%

182 YAML files

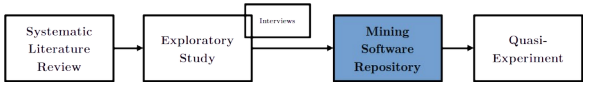
RQ2. How do FLOSS IoT-related projects structure their CI/CD pipeline?



Action	%Steps	%Repositories
actions/checkout	47.17%	100.00%
actions/upload-artifact	12.34%	52.63%
actions/setup-python	8.74%	68.42%
actions/cache	7.84%	31.58%
actions/download-artifact	3.47%	26.32%
codecov/codecov-action	1.03%	15.79%
actions/setup-node	0.90%	10.53%
docker/login-action	0.77%	26.32%
docker/build-push-action	0.77%	21.05%
docker/setup-qemu-action	0.77%	15.79%

75 Actions out of 766 Steps

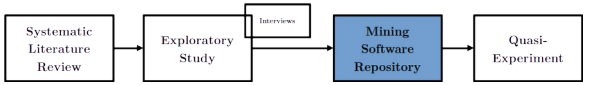
RQ2. How do FLOSS IoT-related projects structure their CI/CD pipeline?



Action category	%Actions
Continuous Integration	26.67%
Utilities	20.00%
Deployment	6.67%
Dependency Management	5.53%
Testing	5.53%
Project Management	5.53%
Code Quality	4.00%
Code Review	2.67%
Open Source Management	2.67%
Chat	2.67%

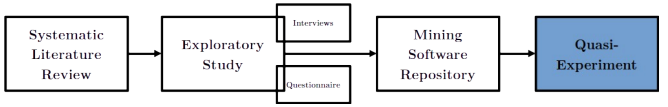
Action category	%Actions
Security	2.67%
Publishing	1.33%
Container CI	1.33%
Uncatecorised	13.33%
Total	**100.00%**

RQ3. What is discussed in FLOSS IoT-related projects communities about the CI/CD pipeline?

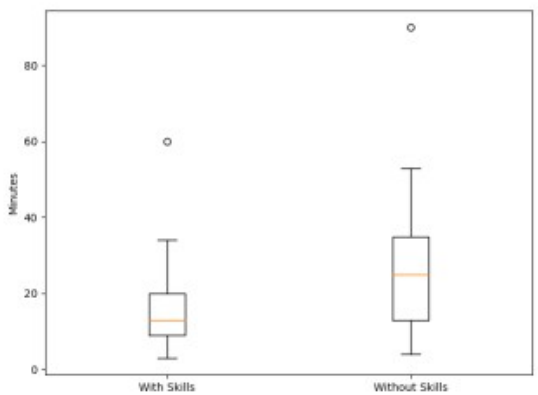


Categories of the Issues	#Issues	%Issues
Difficulties or challenges	83	44.62%
Improvement	44	23.66%
Maintenance	32	17.20%
Request or suggestion	27	14.52%
Total	**186**	**100.00%**

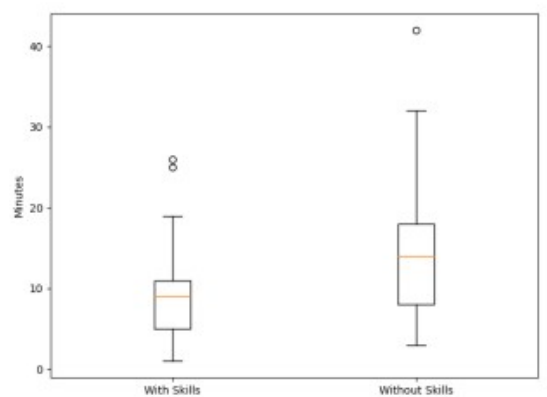
RQ1. How does prior CI/CD knowledge impact the effort to configure, use, and understand a CI/CD pipeline in an IoT embedded project?



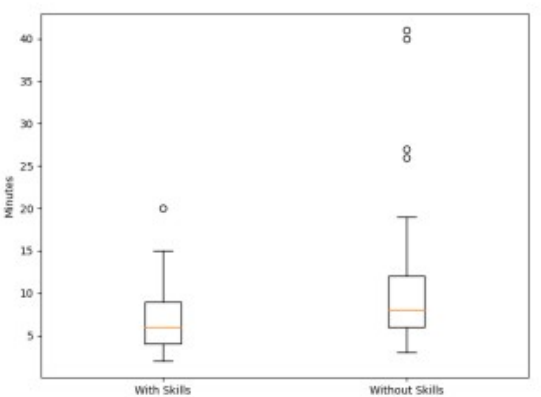
Statistic Table - Mann-Whitney - Time in Minutes							
	With skills			Without skills			
	med	min	max	med	min	max	p-value
Task 1	13	3	60	25	4	90	0.00072
Task 2	9	1	26	12	3	42	0.00220
Task 3	6	2	20	8	3	41	0.00203



Task 1: Set up GitHub Actions

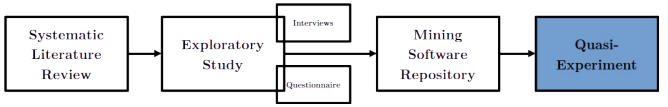


Task 2: Create a PR with a bug



Task 3: Fix a PR

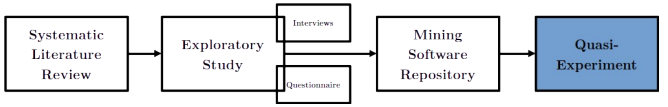
RQ1. How does prior CI/CD knowledge impact the effort to configure, use, and understand a CI/CD pipeline in an IoT embedded project?



Statistic Table - Mann-Whitney - Degree							
	With skills			Without skills			
	mod	min	max	mod	min	max	p-value
Task 1	0	0	4	1	0	4	0.00218
Task 2	1	0	3	2	0	4	0.00022
Task 3	0	0	2	1	0	4	0.00251

Statistic Table - Mann-Whitney - Degree							
	With skills			Without skills			
	mod	min	max	mod	min	max	p-value
File 1	2	0	3	2	1	4	0.00000
File 2	1	0	3	2	0	4	0.02778
File 3	0	0	2	1	0	4	0.00338

RQ2. What were developers' perceptions about the setting, use, and understanding of the CI/CD tasks?



Developers insights	
Don't know GitHub Actions	**Know GitHub Actions**
Novelties (with skills) - 38	Observations (with skills) - 8
Learning (without skills) - 52	
Performance (without skills) - 20	
Doubts (without skills) - 50	
Barriers (without skills) - 53	

Threats to Validity

Threats to Validity

- Construction Validity:
  - SLR: Selection of multiple electronic data sources to ensure comprehensiveness
  - Exploratory Study: Use of a structured interview guide and external validation
  - MSR: Documentation of choices and reasons, recognizing limitations in choosing the GitHub Actions platform
  - Quasi-Experiment: Selection of Software Engineering students as subjects, justified by previous studies

Threats to Validity

- Internal Validity:
  - SLR: Collaborative approaches to increase objectivity in selecting and interpreting studies
  - Exploratory Study: Use of semi-structured interviews to minimize bias
  - MSR: Justifications of methodological choices following recommendations
  - Quasi-Experiment: Document the process and follow guidelines to ensure the validity of inferences

Threats to Validity

- External Validity:
  - SLR: Limitations in the representativeness of selected studies
  - Exploratory Study: Limitations in the sample
  - MSR: Scope limited to IoT-related projects; clear criteria for future studies
  - Quasi-Experiment: Apply appropriate statistical tools and maintain consistency in results

Threats to Validity

- Conclusion Validity:
  - SLR: Consensus among researchers on interpretations
  - Exploratory Study: Collective participation in data analysis and discussion of results
  - MSR: Open coding is used to make raw data available
  - Quasi-Experiment: Detailed presentation of results and encouragement of replication

Conclusion

Summary - SLR

- Practices and Technologies: Various practices and technologies emphasize edge and fog computing
- Infrastructure: Cloud computing and infrastructure as code are emerging trends
- Deployment Strategies: Canary Release, Blue-Green Deployment, and reference architectures
- Challenges Identified: The impact of these challenges, including defining and monitoring QoS metrics, communication between teams, and resistance to agile practices, is significant, underlining the urgency of addressing them

Summary - Exploratory

- DevOps Practices: Generates agility in hardware and firmware development
- Cultural Aspects: Shorter delivery cycles and adaptation to each project's specific demands are needed
- Quality Practices: Create APIs, conduct code reviews, and develop programming guidelines
- Challenges: Incomplete adoption of practices such as clean code and testing by hardware and firmware teams

Summary - MSR

- CI/CD adoption: Mixed picture with variation in workflow adoption
- Tools: The importance of Docker and GitHub Actions is growing
- Gaps: Lack of third-party Actions specific to embedded systems

Summary - Quasi-Experiment

- Impact of Prior Knowledge: Significant differences in effort and understanding between developers with and without prior CI/CD skills
- Benefits of Experience: Experienced developers with prior CI/CD knowledge spend less time and find tasks easier, demonstrating their capability and efficiency
- Beginning Developers: Faced more challenges and spent more time
- Importance of Training: Ongoing training in CI/CD is needed to face emerging technological challenges

Summary - Quasi-Experiment

- Efficiency of GitHub Actions: Praised by experienced developers
- Challenges for Beginners: Interpretation of source code, use of Git and GNU/Linux systems
- Guidance from Researchers: Essential to help overcome obstacles and understand concepts

Summary

- CI/CD Efficiency and Effectiveness:
  - Crucial to the success of modern software projects
- Competitive Advantage:
  - Overcoming barriers and implementing robust CI/CD practices can uniquely position your software projects, providing a significant competitive advantage

Summary

- Investment in Training:
  - It is essential to face current challenges and ensure future success in developing IoT embedded systems
- Recommendation:
  - Look for solutions adapted to the specific needs of each project and promote practical learning

Contributions - SLR

- Identification of Gaps:
  - Future experiments are needed to evaluate the operationalization of CI/CD in IoT embedded systems
- Technology Analysis:
  - A meticulous examination of detailed technology selection requirements, including documentation, active community, and licensing, has been conducted
- DevOps Challenges:
  - Discuss topics, technologies, benefits, and challenges specific to IoT projects, strengthening knowledge for DevOps adoption

Contributions - Exploratory

- Integration of DevOps Practices:
  - Comprehensive view of the problems professionals and organizations face
- Communication and Management:
  - Communication between professionals with different skills and support from top management is important
- Practices and Technologies:
  - Proposing the use of asynchronous communication, continuous monitoring, and careful technology selection, with the promise of improved efficiency and productivity
- Data and Metadata:
  - Use of solutions in production for improvements and continuous learning

Contributions - MSR

- CI/CD adoption:
  - Identification of CI/CD practice in most FLOSS IoT projects, with a predominance of GitHub Actions
- Understanding the user Profiles:
  - Delineating the distinct user profiles for real-time operating system (RTOS) projects and development tools (DT), highlighting their unique needs and challenges
- Pipeline Challenges:
  - Main concerns related to difficulties in using the pipeline and the need for detailed documentation

Contributions - Quasi-Experiment

- Developer Perceptions:
  - Divergences in experiences with CI/CD, highlighting motivation and discovery of new resources
- Technical Challenges:
  - Recognition of pipeline efficiency but technical challenges in executing the steps
- Education and Mentoring:
  - The empowering role of practical and adaptive training approaches, with the guiding hand of mentors, is fundamental in motivating and developing CI/CD skills

Contributions

- Pereira, I.M.; Carneiro, T. G. S.; Figueiredo, E. 2021. A systematic review on the use of DevOps in internet of things software systems. Proceedings of the 36th Annual ACM Symposium on Applied Computing (SAC). pp. 1569-1571. DOI: 10.1145/3412841.3442126.
- Pereira, I.M.; Carneiro, T. G. S.; Figueiredo, E. 2021. Understanding the context of IoT software systems in DevOps. IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT). pp. 13-20. DOI: 10.1109/SERP4IoT52556.2021.00009.
- Pereira, I.M.; Carneiro, T. G. S.; Figueiredo, E. 2021. Main Differences of DevOps on IoT Systems. XXXV Brazilian Symposium on Software Engineering (SBES). pp. 315-319. DOI: 10.1145/3474624.3474630.
- Pereira, I.M.; Carneiro, T. G. S.; Figueiredo, E. 2021. Investigating Continuous Delivery on IoT Systems. XX Brazilian Symposium on Software Quality (SBQS). pp. 1-10. DOI: 10.1145/3493244.3493261.

Contributions

- Analysing the CI/CD Pipeline in FLOSS Repositories of IoT-Related Projects. SUBMITTED TO JUCS - Journal of Universal Computer Science.
- Manipulating a CI/CD Pipeline in an IoT Embedded Project: A Quasi-Experiment. SUBMITTED TO JSEP - Journal of Software: Evolution and Process.

Future Work

Future Work

- Expand the scope of results with additional case studies and experiments to validate tools, people, and solutions
- Evaluate different tools used in IoT embedded projects, focusing on ease of use, hardware integration, and task automation effectiveness
- Develop guidelines and best practices for integrating security into IoT embedded projects, addressing data protection and vulnerabilities
- Conduct experimental research to evaluate the effects of training in CI/CD practices on developers' productivity, quality, and satisfaction in IoT embedded project teams

References

- LUZ, W. P., PINTO, G., BONIFÁCIO, R. "Adopting DevOps in the real world: A theory, a model, and a case study", Journal of Systems and Software, v. 157, pp. 110384, 2019.
- LINDGREN, E., MÜNCH, J. "Raising the odds of success: the current state of experimentation in product development", Information and Software Technology, v. 77, pp. 80-91, 2016.
- JACOBSON, I., SPENCE, I., NG, P.-W. "Is There a Single Method for the Internet of Things? Essence can keep software development for the IoT from becoming unwieldy", Queue, v. 15, n. 3, pp. 25-51, 2017.
- WU, M., LU, T.-J., LING, F.-Y., et al. "Research on the architecture of Internet of Things". In: International Conference on Advanced Computer Theory and Engineering (ICACTE), v. 5, pp. V5-484-V5-487, 2010.
- LEITE, L., ROCHA, C., KON, F., et al. "A Survey of DevOps Concepts and Challenges", ACM Computing Surveys, v. 52, n. 6, 2019.
- ZHANG, D., CHAN, C. C., ZHOU, G. Y. "Enabling Industrial Internet of Things (IIoT) towards an emerging smart energy system", Global Energy Interconnection, v. 1, n. 1, pp. 39-47, 2018. ISSN: 2096-5117.
- KUMAR, R., AGRAWAL, N. "Analysis of multi-dimensional Industrial IoT (IIoT) data in Edge-Fog-Cloud based architectural frameworks : A survey on current state and research challenges", Journal of Industrial Information Integration, v. 35, pp. 100504, 2023. ISSN: 2452-414X.

Acknowledgements

THANK YOU!

Questions?