Time	Type	Person	Title
9:00	Welcome	Mehrdad Sabetzadeh, Maleknaz Nayebi	Welcome and plan of the day
9:10	Keynote	Jinqiu Yang	Beyond Accuracy: Engineering Reliable Machine Learning Systems
10:00	Break		
10:30	New Faculty	Jessie Galasso	Representing and managing collections of similar SE artefacts
10:50	New Faculty	Cristiano Politowski	Advancing Automated Game Testing through Open-source Data Curation and Pattern Discovery
11:10	New Faculty	Ali Ayub	Long-Term Real-World Personalization for Autonomous Systems
11:30	New Faculty	Taher Ghaleb	AI4CI: Breaking Barriers, Unlocking Solutions
11:50	New Faculty	Hamid Mcheick	Design context aware healthcare framework
12:10	Postdoc	Aren A. Babikian	Safety Assurance of Automotive Systems in the Presence of Change
12:30	Lunch		
13:30	PhD talk	Mohammad Hossein Amini	Bridging the Gap between Real-world and Synthetic Images for Testing Autonomous Driving Systems
13:40	PhD talk	Mahtab (Mattie) Nejati	Understanding the Implications of Changes in Build Systems
13:50	PhD talk	Nimmi Rashinika Weeraddana	Dependency-Induced Waste in Continuous Integration: An Empirical Study of Unused Dependencies in the npm Ecosystem
14:00	PhD talk	Shayan Noei	An Empirical Study on Release-Wise Refactoring Patterns
14:10	PhD talk	Pouya Fathollahzadeh	Towards Refining Developer Questions using LLM-Based Named Entity Recognition
14:20	PhD talk	Hamed Taherkhani	Hallucination-aware LLM-generated test case validation
14:30	PhD talk	Yiping Jia	Automated Function Synthesis with LLM Supported Agents
14:40	PhD talk	Jiho Shin	Domain Adaptation for Code Model-Based Unit Test Case Generation
14:50	PhD talk	Nima Shiri Harzevili	Checker Bug Detection and Repair in Deep Learning Libraries
15:00	Break		
15:30	Poster session starts		
16:55	Closing	Mehrdad Sabetzadeh, Maleknaz Nayebi	Next CSER, awards….

Timestamp	Please indicate your full name	Please indicate the name of your supervisor
10-10-2024 11:00:25	Elmira Onagh	Dr. Maleknaz Nayebi
10-24-2024 22:07:39	Elim Lemango	Dr. Maleknaz Nayebi
10-23-2024 9:23:10	Nimmi Rashinika Weeraddana	Dr. Shane McIntosh
10-25-2024 11:12:33	Mohammad Hossein Amini	Dr. Shiva Nejati
10-18-2024 12:51:59	Huizi Hao	Dr. Yuan Tian, Dr. Ahmed E. Hassan
10-22-2024 19:41:57	Hamed Taherkhani	Hadi Hemmati
10-24-2024 14:28:24	Melika Sepidband	Hadi Hemmati
10-22-2024 15:15:54	Alireza Daghighfarsoodeh	Hung Viet Pham
10-23-2024 14:05:08	Chung-Yu Wang	Hung Viet Pham
10-22-2024 15:16:17	Faiz Ahmed	Maleknaz Nayebi
10-22-2024 15:41:28	Sharuka Promodya Thirimanne	Maleknaz Nayebi
10-25-2024 10:45:14	Ramin Sharifi	Maleknaz Nayebi - Zhen Ming Jiang
10-9-2024 20:10:53	Fares Hamouda	Marios Fokaefs
10-25-2024 21:07:31	Hasti Ghaneshirazi	Marios Fokaefs
10-25-2024 23:52:23	Hashim Khan	Marios Fokaefs
10-24-2024 15:22:14	Aren A. Babikian	Marsha Chechik
10-23-2024 7:08:51	Mahtab (Mattie) Nejati	Prof. Shane McIntosh
10-21-2024 10:04:27	Oluwafemi Odu	Professor Alvine Belle Boaye
10-22-2024 14:35:07	Gengyi Sun	Shane McIntosh
10-8-2024 19:56:35	Nima Shiri Harzevili	Song Wang
10-14-2024 0:00:25	Reem Aleithan	Song Wang
10-13-2024 13:28:14	Kimya Khakzad Shahandashti	Song Wang, Alvine Boaye Belle
10-7-2024 12:56:29	Jiho Shin	Song Wang, Hadi Hemmati
10-24-2024 16:24:44	Pouya Fathollahzadeh	Ying Zou
10-25-2024 16:30:39	Yiping Jia	Ying Zou
10-26-2024 20:50:06	Shayan Noei	Ying Zou

Please indicate the name of your current institute	Please indicate the email address we can best c
York University	eonagh@yorku.ca
York University	elim17@my.yorku.ca
University of Waterloo	nrweeraddana@uwaterloo.ca
University of Ottawa	mh.amini@uottawa.ca
Queen's University	huizi.hao@queensu.ca
York University	hamedth@yorku.ca
York University	sepidband.m@gmail.com
York University	aliredaq@yorku.ca
York University	cywang14@yorku.ca
York University	faiz5689@my.yorku.ca
York University	sharukat@yorku.ca
York University	ramins@yorku.ca
York University	hamoudaferes@gmail.com
York University	hastighsh@gmail.com
York University	hashimahmedkhan2002@gmail.com
University of Toronto	babikian@cs.toronto.edu
University of Waterloo	mattie.nejati@uwaterloo.ca
York University	olufemi2@yorku.ca
University of Waterloo	gengyi.sun@uwaterloo.ca
York University	nshiri@yorku.ca
York University	reem1100@yorku.ca
York University	kimya@yorku.ca
York University	jihoshin@yorku.ca
Queen's University	22pf2@queensu.ca
Queen's University	yiping.jia@queensu.ca
Queen's University	s.noei@queensu.ca

Please indicate your current position/Program	Please indicate the year of your studies MsC/Phl
Computer Science	2
Research Assistant	N/A
PhD	3
Ph.D. in Electrical Engineering and Computer Sc	2022 - 2026
MsC in Computing, area of study in Software En	Currently in second year of MsC. Short Bio: Huizi
PhD in computer science	PhD
Computer Science	PhD
Research Assistant - Student	MsC
Computer Science	second year MSc
Gradtuate Student/Computer Science	MSc
M.Sc. Computer Science	Recently Graduated
PhD EECS	First Year
Student	2nd year
Computer Science	4th year undergraduate
Research Assistant	graduated from bachelors
Postdoctoral fellow	postdoc 1
PhD Candidate	Release Pipelines, Build Systems, Mining Softwa
Master's Student/ Electrical Engineering and Cor	MsC/ Second Year
Ph.D	1st
Ph.D. candidate	5th PHD
Computer Science	MsC
Researcher	Graduated with MSc
PhD student	4th year
PhD student / Electrical and Computer Engineeri	Third
Ph.D. student	3rd year
PhD Student	5

Please provide the title of your talk/poster	Please provide a 250 words abstract for your pro
Extension Decisions in Open Source Software	GitHub Marketplace, as an open-source software
Anonymity in Developer Communities: Insights	The paper is structured around two primary studi
Dependency-Induced Waste in Continuous In	Modern software systems are increasingly dependent upon code from external packages (i.e., dependencies). Building upon external
Bridging the Gap between Real-world and Syr	Deep Neural Networks (DNNs) for Autonomous D
I would like to give a talk, with the title: "Under	ChatGPT has significantly impacted software dev
Hallucination-aware LLM-generated test case	The generation of test cases using Large Langu
TBA	TBA
Code Generation Benchmark for deep learnin	The emergence of deep learning (DL) has revolu
Application and optimization of prompt engine	Large Language Models (LLMs) have demonstra
Context Aware Recognition from Text and Coc	Developers frequently share screenshots when p
Retrieval-Augmented Editing of TensorFlow Al	With the rapid growth of machine learning librarie
Diagnostic Complexities in Modern UAV Syste	The rapid development of unmanned aerial vehic
DMBench: Load Testing and Benchmarking T	Data migration refers to the set of tasks around t
DMML: A Machine-learning Performance Mode	Data migration between systems with varying cor
Use of Large Langauge Models to generate S	Software architecture plays a critical role in defini
Safety Assurance of Automotive Systems in t	In recent years, road vehicles have become incre
Understanding the Implications of Changes in	The maintenance of build systems imposes a cor
SmartGSN: A Generative AI-powered Online T	Developing industry-wide standards and making
RavenBuild: Context, Relevance, and Depend	Continuous Integration (CI) is a common practice
Checker Bug Detection and Repair in Deep Le	Checker bugs in Deep Learning (DL) libraries are
SWE-Bench+: Enhanced Coding Benchmark I	Large Language Models (LLMs) are increasingly
Program Slicing in the era of Large Language	Program slicing is a critical technique in software
Domain Adaptation for Code Model-Based Un	Recently, deep learning-based test case generat
Towards Refining Developer Questions using	In software engineering chatrooms, communicati
Automated Function Synthesis with LLM Supp	LLMs have been used to conduct various coding
An Empirical Study on Release-Wise Refactori	Refactoring is a technical approach to increase th

posed talk/poster
platform, is rapidly evolving, with an average annual increase of 41.23\% in new tools by new or e
es: an interview study with 34 early-career developers and a mining study analyzing 130,000 deve

riving Systems (ADS) are typically trained on real-world images and tested using synthetic images f
velopment practices, providing substantial assistance to developers in various tasks, including codi
age Models (LLMs) is critical in ensuring the correctness of code in automated software developme

tionized various fields, enabling significant advancements in areas such as computer vision, natura
ated impressive abilities across various tasks, particularly in software engineering. However, they str
osting questions on Q&A platforms, yet the responses they receive rarely incorporate visual cues,
s, accurate and detailed API documentation is essential to help developers effectively use these t
cles has introduced unique technical challenges , creating a complex diagnostic landscape for inve
ransferring data over a network between two systems, either homogeneous or heterogeneous, an
nfigurations and data characteristics is critical for enterprises, often requiring significant time, cost, a
ng the structure and behavior of a system. The process of architecting involves understanding the
easingly software-driven: vehicles commonly integrate numerous software components, from infotai
nsiderable overhead on software development. Since automated quality assurance methods are ra
sure producers of mission-critical systems comply with them is crucial to foster consumer acceptanc
adopted by modern software organizations. It plays an especially important role for large corporat
critical yet {not} well-explored. These bugs are often concealed in the input validation and error-ch
used in Software Engineering (SE) for coding assistance. To assess LLMs in practical coding task
engineering, enabling developers to isolate relevant portions of code for tasks such as bug detect
tion approaches have been proposed to automate the generation of unit test cases. In this study,
on is often hindered by imprecise questions that cannot be answered. Existing research in this are
tasks, such as code completion, code repairs and code generation. However, LLM's ability in gen
he internal quality of software without altering its external functionalities. With the increased adopti

existing providers. In such a dynamic software ecosystem (SECO), making informed decisions abou
loper profiles. The interview study investigates developers' definition of anonymity, the activities for

from simulators. This approach results in training and test datasets with dissimilar distributions, whic
ng, testing, and debugging. Despite its widespread adoption, the impact of ChatGPT as an assista
ent. However, many LLM-generated test cases are invalid, leading to reduced efficiency and potent

al language processing, and more. However, many DL systems are developed by domain experts ra
ruggle with task-oriented prompts and generating accurate and reliable code due to a lack of specif
forcing readers to interpret details from text alone. Inspired by the growing trend of visual communi
ools. Modern software frequently undergoes updates—major, minor, and patches—to improve stab
estigators, as malfunctions often blur the line between software bugs and hardware failures, making
d the potential reformatting of this data. Combined with large volumes of data, resource constraints
and resource allocation. To optimize this process, we developed a machine learning-based approac
problem, finding appropriate solutions, and evaluating the final architecture to ensure it addresses
nment systems to autonomous driving features. While the inclusion of such software-intensive featu
arely applied to build specifications, the importance of the role peer code review plays in the mainte
e. Producers of such systems can rely on assurance cases to demonstrate to regulatory authorities
ions like Ubisoft, where thousands of build jobs are submitted daily. Indeed, the cadence of develo
hecking code of DL libraries and can lead to silent failures, incorrect results, or unexpected program
s, Carlos et al. introduced the SWE-bench dataset, consisting of 2,294 GitHub issues and pull requ
tion, code comprehension, and debugging. In this study, we investigate the application of large lan
we leverage Transformer-based code models to generate unit tests with the help of Domain Adapt
a primarily focuses on improving question clarity using natural language processing techniques. Ho
erating code for complex tasks is still limited.We aim to enhance code generation performance thro
on of continuous integration and development (CI/CD), refactoring activities may vary within and ac

t market entry and product extension is critical for success and requires careful consideration. GitHu
which they prefer anonymity, and their engagement with privacy policies. The study also assesses

ch can potentially lead to erroneously decreased test accuracy. To address this issue, the literatur
nt in collaborative coding remains largely unexplored. To fill this gap, we analyze a dataset of deve
tial errors in software systems. This work introduces a novel framework designed to automatically va

ather than software developers, creating coding challenges due to the complexity of DL workflows.
fic prior knowledge and prompt engineering challenges. Existing approaches, like PAL, use code ge
ication and advancements in image processing, our study explores how effectively images can be p
bility and performance, but documentation often lags behind, leading to more documentation-relate
it difficult for investigators to pinpoint the root cause of flight anomalies. Understanding and reproc
s and variety in data models and formats, data migration can be critical for enterprises, as it can con
ch to estimate transfer times for migrations based on system configurations and data properties. Thi
the system's requirements. Traditionally, software architecting has been a manual and complex tas
ures enhances the driving experience from the user's perspective, it also introduces significant safe
nance of build systems is amplified. Yet prior work shows that the review process for build systems
s how they have complied with such standards to help prevent system failure. Assurance cases are
pment progress is constrained by the pace at which CI services process build jobs. To provide faste
behavior in DL applications. Despite their potential to significantly impact the reliability and perform
uests from 12 Python repositories. However, a systematic evaluation of the dataset's quality has be
guage models (LLMs) to both static and dynamic program slicing, with a focus on Java programs. W
ation (DA) at a project level. Specifically, we use CodeT5, a relatively small language model trained
wever, existing techniques often overlook contextual nuances specific to software development, su
ugh the collaboration of different agents. A feedback loop refines the generated code by categoriz
ross different releases and be influenced by various release goals. However, there is a lack of exist

ıb Marketplace offers an environment where developers can openly share tools for automating work
whether presenting privacy policies in a format based on contextual integrity improves developers' u

e suggests applying domain-to-domain translators to test datasets to bring them closer to the trainir
lopers' shared conversations with ChatGPT in GitHub pull requests (PRs) and issues. In our first stu
lidate test cases generated by LLMs by leveraging token probabilities. We evaluate our approach u

Large Language Models (LLMs), like GPT-4o, have emerged as promising tools to assist in DL cod
eneration to address this but depend heavily on manually crafted prompt templates and examples,
erceived and interpreted by state-of-the-art methods and Large Language Models (LLMs).To this e
d issues on GitHub and questions on Stack Overflow. Further analysis on the Stack Overflow questi
lucing these incidents presents significant challenges in the unmanned aerial vehicle (UAV) industry
sume a significant amount of time, incur high costs, and pose a significant risk if not executed corre
is project extends the DMBench tool, which benchmarks data migration performance, by incorporati
sk, heavily reliant on the architect's expertise and knowledge. This manual process presents several
ty assurance challenges. A key particularity of automotive systems is the significant level of change
suffers from a lack of build experts and effective tooling.To support the understanding of changes t
mainly used in safety-critical areas (e.g., automotive, aerospace) to deal with high-risk concerns an
er CI feedback, recent work explores how build outcomes can be anticipated. Although early results
ance of DL-enabled systems built with these libraries, checker bugs have received limited attention.
en lacking. This paper presents an empirical analysis of SWE-bench, focusing on the performance c
We evaluate the performance of four state-of-the-art LLMs—GPT-4o, GPT-3.5 Turbo, Llama-2, and
on source code data, and fine-tune it on the test generation task. Then, we apply domain adaptat
ch as programming languages, libraries, and versioning details. In this paper, we propose SENIR: a
ting execution errors (e.g., compile errors, test failures, timeouts) and iteratively improving the output
ing work to understand how practitioners distribute their refactoring activities in a release and their i

cflows in open-source software. However, our preliminary analysis of this ecosystem revealed a signi
understanding. We found that early-career developers define anonymity as the state of not sharing

ng datasets. However, translating images used for testing may unpredictably affect the reliability, eff
dy, we manually examined the content of the conversations and characterized the dynamics of dev
using nine test suites generated from three datasets across three LLMs. Our results suggest that to

e generation, offering potential solutions to these challenges. Despite this, current benchmarks, suc
often producing inaccurate results. To overcome these limitations, we propose two novel strategies
end, we introduce CORTEx, a novel method designed to identify programming context and extract b
ions related to TensorFlow API documentation revealed that majority of these questions arise from i
y. Current methods for analyzing flight incidents heavily depend on expert knowledge and manual in
ectly. The ability to accurately and effectively predict these challenges and plan for proper resource,
ng predictive modeling.Our method involved gathering detailed data from migration experiments, inc
l challenges, such as inconsistency, inefficiency, and the risk of overlooking essential design patterr
and variability that they encounter, which makes their safety assurance particularly challenging. On
o build specifications (a key stage in the review process), we propose BCIA—an approach to summ
d show to stakeholders that such systems are safe according to domain-specific criteria. Assurance
show plenty of promise, the distinct characteristics of Project X—a AAA video game project at Ubis
The complex nature of DL libraries, combined with the rapid pace of their development, has left a g
of SWE-Agent + GPT-4, which led the SWE-bench leaderboard during our study.Our analysis revea
Gemma-7B—byleveraging advanced prompting techniques, including few-shot learning and chain-o
ion to each target project data to learn project-specific knowledge (project-level DA). We use the Me
an approach for S oftwarE -specific Named entity recognition, Intent detection, and Resolution statu
t.The Repairer Agent fixes code based on error types and messages, while the Mentor Agent summ
mpact. Therefore, we empirically study the frequent release-wise refactoring patterns in 207 open-s

ificant functional redundancy among the provided tools in this SECO.\\We employed a graph-based
identifiable information such as name, location, picture, and professional background. The mining s

fectiveness and efficiency of the testing process. Hence, this paper investigates the following ques
velopers' sharing behavior. The main observations are: (1) Developers seek ChatGPT's assistance a
ken probabilities are a reliable indicator for distinguishing between valid and invalid test cases, offer

ch as DS-1000, are limited, focusing primarily on small DL code snippets related to pre/post-process
: TITAN and PET-Select. TITAN enhances LLM performance on task-oriented prompts by employing
both code and text from images posted on Q&A platforms. CORTEx works by segmenting the image
inadequate examples and documentation ambiguity. To address this, we propose DocChameleon,
nterpretation of extensive amounts of data, making it particularly challenging to definitively determin
time and budget allocation is vital for the proper execution of data migration. In this work, we introc
cluding system features such as RAM, CPU, disk size, and the number of data streams, along with
ns and non-functional requirements. These mistakes during the design phase of the architecture ca
one hand, automotive systems interact with a perpetually changing traffic environment, which must
arize the impact of changes to build specifications across the build configuration space. BCIA trave
cases are usually very large documents that can span several hundred pages. Their management
oft—present new challenges for build outcome prediction. In the Project X setting, changes that do
gap in understanding how these bugs manifest, propagate, and affect the overall robustness of AI
led significant issues: 32.67% of successful patches involved "solution leakage," where solutions w
of-thought reasoning. Using a dataset of 100 Java programs derived from LeetCode problems, our
ethods2test dataset to fine-tune CodeT5 for the test generation task and the Defects4j dataset for
s detection to analyze developer chatroom conversations. By leveraging the DISCO dataset, a con
narizes successful fixes and identifies patterns to guide future repairs. These examples are integrate
ource Java projects and their characteristics. Then, we analyze how these patterns and their transit

l approach to analyze the functional relationships between 6,983 ``Continuous Integration" Actions
study, motivated by the findings of the interview study, explores the extent to which developers sha

tions in the context of ADS: \emph{Could translators reduce the effectiveness of images used for A
across 16 types of software engineering inquiries. The most frequently encountered inquiry categori
ing a robust solution for improving the reliability of LLM-generated test cases in software testing.

sing tasks and lacking comprehensive coverage across the full DL pipeline, including different DL ph
g a universal, zero-shot learning approach that eliminates the need for detailed task-specific instruct
into various sections and identifying text within specific areas, significantly improving accuracy. Usi
an automated TensorFlow API documentation augmentation method using Large Language Mode
e whether failures stem from software bugs or hardware malfunctions, while the overall analysis pro
luce the concept of load testing and benchmarking for data migration to allow decision-makers for h
data characteristics like total size, row count, and compression type. After cleaning and engineering
n show up later, imposing high maintenance costs.This exhibit explores the potential of Large Lang
t be considered when defining system-level safety assurance and testing approaches, as prescribe
rses the paths through which data and control flow in the prior and updated versions of the build sy
(e.g., creation, conversion) is usually manual, which can be time-consuming, tedious, and prone to
not modify source code also incur build failures. We also observe that the code changes that have
systems.To fill this gap, we present the first comprehensive study of DL checker bugs in two widely-
ere provided in the issue reports or comments. Additionally, 31.08% of the patches were questional
experiments reveal that GPT-4o performs the best in both static and dynamic slicing across other LL
project-level domain adaptation and evaluation. We compare our approach with (a) CodeT5 fine-tur
nprehensive dataset of tagged chatroom conversations, SENIR achieves an 86% F-score for entity
ed into a Retrieval-Augmented Generation (RAG) system, which retrieves similar cases to assist in c
ions affect code quality. We identify four major release-wise refactoring patterns: early active, late a

and 3,869 providers. We investigated the birth of functionalities and their evolutionary trajectory ov
re identifying information across social coding platforms and how easily their professional profiles ca

ADS-DNN testing and their ability to reveal faults in ADS-DNNs? Can translators result in excessive t
es include code generation, conceptual questions, and how-to guides. (2) Developers frequently er

ases and input types.To address this, we introduce \tool, a novel benchmark dataset designed for
tions. Using step-back prompting and chain-of-thought prompting, TITAN improves code generation
ng Stack Overflow as our case study, we compare CORTEx's performance against Google Vision O
els (LLMs) and the Retrieval-Augmented Generation (RAG) technique. DocChameleon generates ex
cess remains time-consuming and potentially inconsistent.A particularly challenging aspect is the re
igher efficiency and effectiveness when planning for such tasks. Our framework aims for extensibilit
features, we applied a custom preprocessing pipeline to handle missing values, scale numerical fe
guage Models (LLMs), like ChatGPT, to assist software architects in designing software to overcome
d by existing international safety standards (e.g. ISO 26262). On the other hand, the software withi
ystem to generate an Impact Knowledge Graph (IKG), which describes the impact of the change ac
error. Several tools have been developed to assist argument developers in managing assurance c
e an impact that crosses the source-data boundary are more prone to build failures than code chan
used DL libraries, i.e., TensorFlow and PyTorch. Initially, we automatically collected a dataset of 2,4
ble due to weak test cases. Filtering out these problematic cases dropped SWE-Agent + GPT-4's re
LMs, achieving an accuracy of 60.84% and 59.69%, respectively. Our results also show that the LL
ned on the test generation without DA, (b) the A3Test tool, and (c) GPT-4 on five projects from the
recognition, a 71% F-score for intent detection, and an 89% F-score for resolution status detection
ode correction using methods like longest sequential matching and k-nearest neighbors.This collab
ctive, steady active, and steady inactive. We find that adopting the late active pattern—characterize

er time by mining the version control history and different releases of the actions and identified earl
n be retrieved using this information. The findings reveal that developers have varying degrees of c

ime overhead during simulation-based testing?} To address these questions, we consider three dor
ngage with ChatGPT via multi-turn conversations where each prompt can fulfill various roles, such a

functional-level DL code generation, incorporating three key categorizations: DL phases (e.g., pre-p
through a streamlined process, refining scripts via post-processing to retrieve accurate results. PET
CR, LLMs like ChatGPT-4 and Gemini 1.5 Pro, and state-of-the-art methods such as PSC2code an
xecutable code examples to tackle the lack of examples, clarifies ambiguities with detailed explanati
production of flight incidents in Software-In-The-Loop (SITL) simulation environments. This process
y and customizability to enable the execution of a greater variety of tests. Here, we present a proto
atures, and encode categorical variables.We trained multiple regression models, including Decision
these problems. We will present our progress in developing a collaborative process where LLMs ca
n the automotive system itself is subject to variability, often in the form of over-the-air update, which
ross the build configuration space. We develop BuiScout—a prototype implementation of BCIA for
ases. However, most of these tools primarily focus on a single aspect of assurance case managem
ges that do not impact data files. Since such changes are not fully characterized by the existing se
-18 commits from TensorFlow and PyTorch repositories on GitHub from Sept. 2016 to Dec. 2023 us
esolution rate from 12.47% to 3.97%. Similar data quality concerns were found in SWE-bench Lite a
Ms we experimented with are yet to achieve reasonable performance for either static slicing or dyna
Defects4j dataset. The results show that tests generated using DA can increase the line coverage b
, outperforming existing models such as the one by [38] (78% F-score for NER). SENIR achieves a
orative, feedback-driven approach shows promise in improving the performance of LLM-based code
ed by increasing refactoring activities as the release approaches—leads to the best code quality. W

y adopters. Understanding the dynamics of the GitHub Marketplace is crucial for developers
data sharing across patterns on different platforms. Our findings from the mining

main-to-domain translators: CycleGAN and neural style transfer, from the liter
s unveiling initial tasks, iterative follow-up, and prompt refinement. (3) In collaborative codin

processing, model construction, training), input types (e.g., tabular, image, text), and machine le
T-Select, a PET-agnostic model, tackles prompt engineering challenges by selecting the most suitab
d CodeMotion in interpreting code-related images.Our results show that CORTEx outperforms existi
ions, and provides relevant YouTube tutorials and Stack Overflow posts for additional API knowledg
requires recreation of flight conditions, environmental parameters, and system st
type architecture, a roadmap of how the development of such a platform sh
Tree, Random Forest, Gradient Boosting, and XGBoost, and evaluated them using metrics su
an generate software architectures from textual system requirements through carefully
raises questions about the impact of said updates on the safety assurance procedur
CMake-based build systems. We use BuiScout to evaluate our approach through an empirical stud
ent, such as its automatic creation, and largely depend on the Model-Driven Engineering (MDE) me
t of features for build outcome prediction, state-of-the-art models tend to underperform. T
ing specific keywords related to checker bugs. Through manual inspection,
nd SWE-Bench Verified variants.To address these issues, we developed SWE-bench+, a refined d
amic slicing. Through a rigorous manual analysis, we developed a taxonomy of root causes and failu
by 18.62%, 19.88%, and 18.02% and mutation score by 16.45%, 16.01%, and 12.99% compared t
15% improvement in accuracy for entity recognition and a 10% improvement in predicting resolution
e generation for complex tasks.
We observe that as projects mature, refactoring becomes more acti

ble Pr
ing methods and LLM
ge.We performed a comprehe

y of 10
thodolog

ataset consis
ure location
o the abo
n outco

Timestamp	Indicate you full name	Indicate your current institute
10/3/2024 15:59:22	Ali Ayub	Concordia University
10/3/2024 16:49:41	Hamid Mcheick	University of Quebec at Chicoutimi
10/4/2024 10:48:53	Cristiano Politowski	Ontario Tech University
10/17/2024 11:18:36	Jessie Galasso	McGill University
10/22/2024 14:39:15	Taher Ghaleb	Trent University

Indicate your rank	Indicate the email we can best contact you
Assistant Professor	ali.ayub@concordia.ca
Full professor	hamid_mcheik@uqac.ca
Assistant Professor	cristiano.politowski@ontariotechu.ca
Assistant Professor	jessie.galasso-carbonnel@mcgill.ca
Assistant Professor	taherghaleb@trentu.ca

Indicate tentative title of your talk
Long-Term Real-World Personalization for Autonomous Systems
Design context aware healthcare framework
Advancing Automated Game Testing through Open-source Data Curation and Pattern Discovery
Representing and managing collections of similar SE artefacts
AI4CI: Breaking Barriers, Unlocking Solutions