Just in Time Software Maintenance through Clone Detection and Mitigation

Palash R. Roy, Kevin Schneider,
Banani Roy
Dept. of Computer Science
University of Saskatchewan

UNIVERSITY OF SASKATCHEWAN

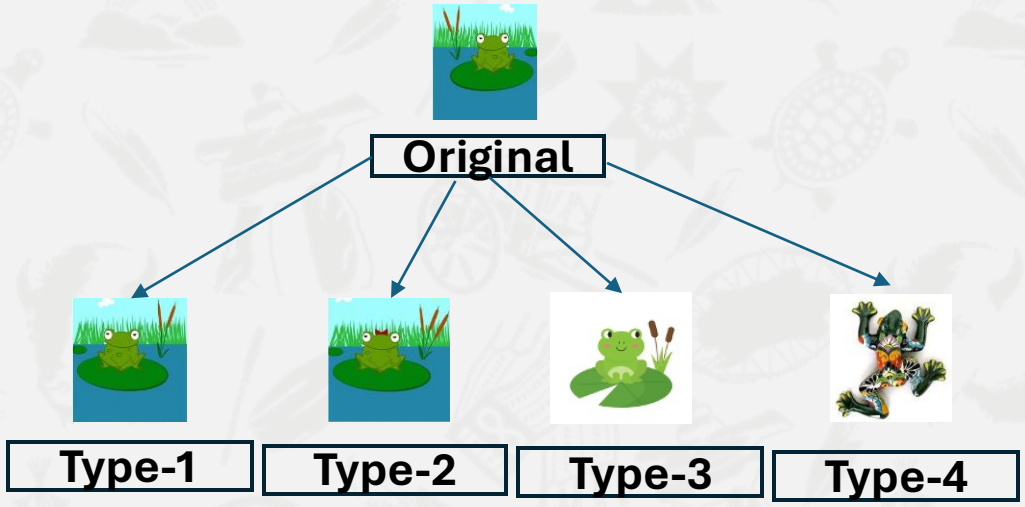
BE WHAT THE WORLD NEEDS

SR LAB Software Research Lab

iSE LAB INTERACTIVE SOFTWARE ENGINEERING LAB

ICSME 2025 Auckland, New Zealand

Similar Codes/Clones



Semantic Clones

```python
def factorial(n):
    x = n
    for i in range(1,n,1):
        x = x*(n-i)
    return x
```

```python
def factorial(n):
    if n <= 1:
        return 1
    else:
        return n * factorial(n-1)
```

Cross-Language Clones

```java
// Java
public static boolean longerTF (boolean [] guess) {
    int variableTrue = 0;
    int variableFalse = 0;
    for (int x = 0;
    x < guess.length; x ++) {
        if (guess [x]) {
            variableTrue ++;
        } else {
            variableFalse ++;
        }
    }
    return variableTrue >= variableFalse;
}
```

```python
// Python
def longerTF(guess):
    variableTrue = 0
    variableFalse = 0
    for x in range(len(guess)):
        if (guess[x]):
            variableTrue += 1
        else:
            variableFalse += 1
    return variableTrue >= variableFalse
```

Code Clones: What & how Common

- Duplicated or near duplicated code fragments.
- 7-24% and sometimes up to 50%.
- Shortcut now, Dept later.



Impacts of Clone

Positive Impacts *[Krinke WCRE 2007, Krinke SCAM 2008, Krinke, IWSC 2011, Hotta et al. IWPSE 2010, Gode and Harder CSMR 2011, Kapser and Godfrey EMSE 2008, Alam et al. ICSME 2023, Roy et al. IWSC 2023, Mondal et al EMSE 2017]*

- Increased productivity by code reuse.
- Easing Program Comprehension.
- Avoiding risks by reuse of mature and tested code.

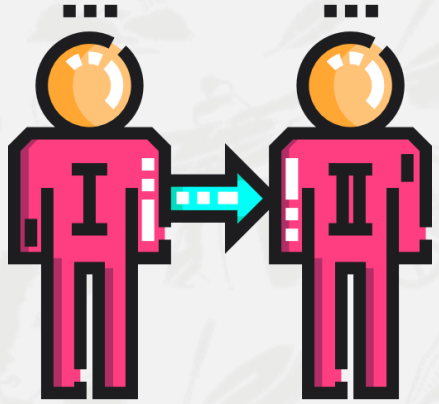


Negative Impacts *[Lozano and Wermelinger, ICSM 2008 , Lozano and Wermelinger, IWSC 2010, Juergens et al. ICSE 2009, Barbour et al. ICSM 2011, Mondal et al. ACR 2012, SAC 2012]*

- Hidden bug propagation.
- Inconsistent change.
- Higher change proneness.
- Larger codebase (code bloating).
- Increased maintenance efforts.

Why Clones Matter?

- Break DRY principle.
- Cause bug propagation.
- Increase maintenance cost and regressions.
- A classic "bad smell" in software.



The AI era: A double edged sword

- 92% North American Developers are using AI Coding Assistants.
- Boost productivity (Co-pilot, Cursor e.t.c.)
- AI often duplicates existing logic.
- 4x increase in duplicated code.

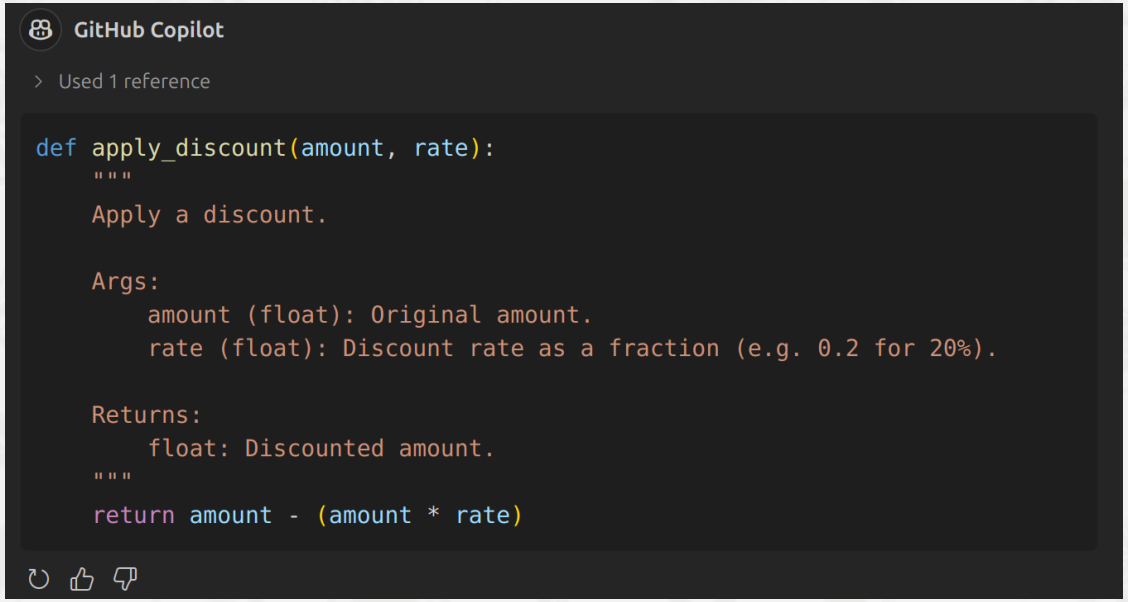


Human vs AI

Existing function

```
165
166 def calculate_discount(price, discount_rate):
167     return price - (price * discount_rate)
168
```

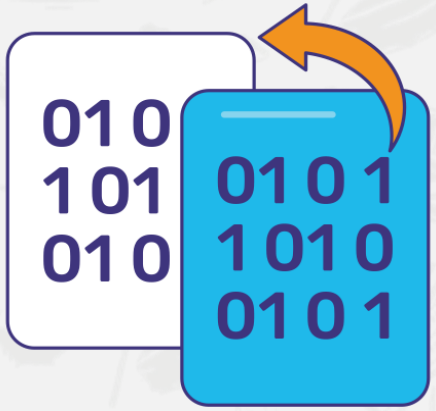
AI generated



Towards Just-In-Time, Inclusive Clone Refactoring

State of Practice: Clone Management Today

- Reactive approaches dominate
- Tools run after clones introduced
- Often addressed during reviews or maintenance
- Result: Clones already propagated



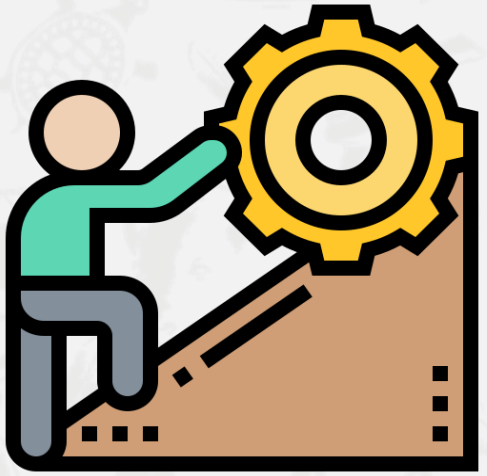
Existing Tools & Gaps

- Miss AI clones
- Weak IDE integration
- Lack prioritization/context



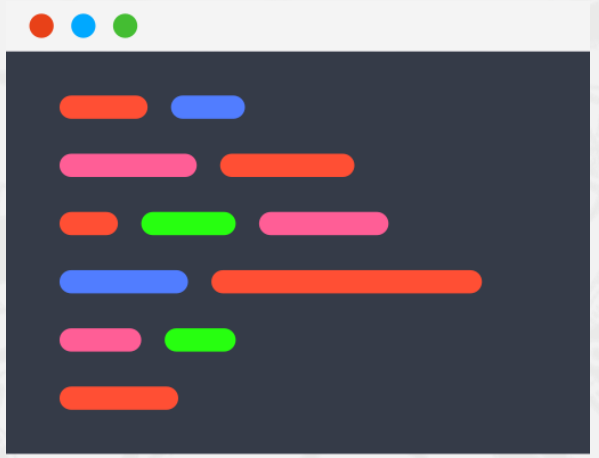
New Challenges: AI & Clone Management

- AI Suggestions can mislead
- Legal and licensing risks
- Human trust and adoption

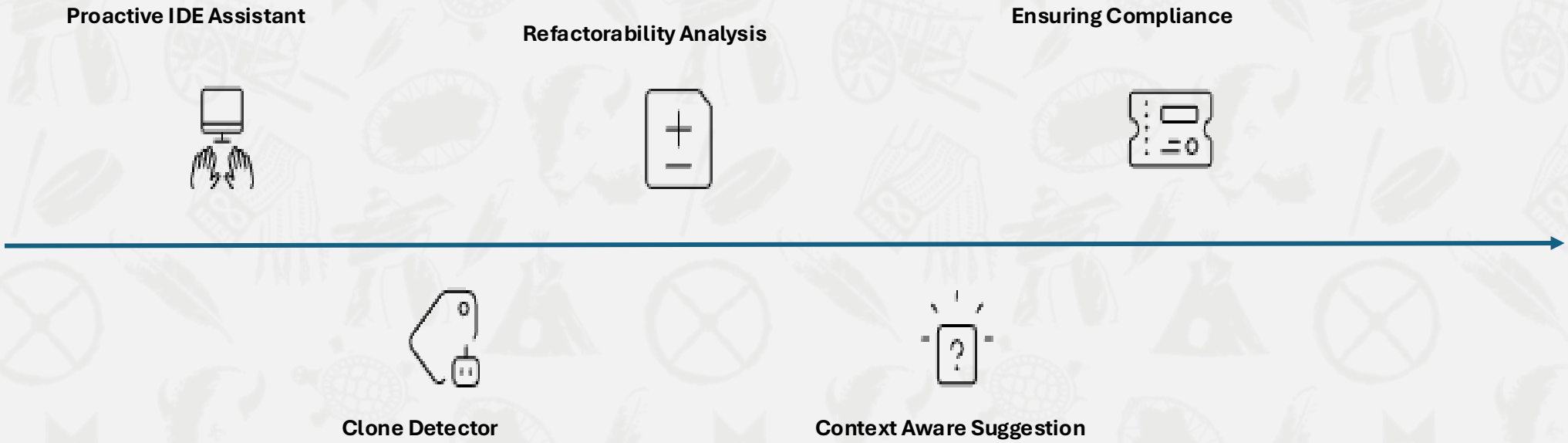


Vision: An IDE assistant that

- Detects clones as you code.
- Provide suggestions to mitigate the clone problem.
- Checks legal/licensing safety.
- Includes developer in the loop

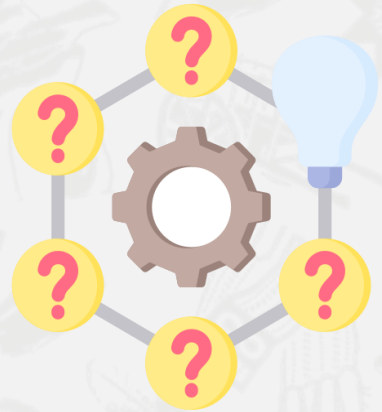


Proposed Solution: AI-Powered Just-In-Time Clone Refactoring



Research Questions

- How can proactive clone management be aligned with developers practices and integrated into modern IDE workflows?
- How do AI enhance automated clone detection, refactoring and what are the key limitations of these approaches?
- What benefits does proactive clone management provide in terms of software quality and maintainability?
- How can clone management systems be extended to detect and mitigate licesnse and intellectual property risks in code reuse?
- What role should developer feedback and explainability play in fostering trust in AI driven refactoring tools?



Evaluation Plan

- Quantitative:
  - Workflow Impact: Deploy IDE plugin in controlled projects.
  - Benchmarking: Rate of invalid or hallucinated suggestions.
  - License Checks: validate detection of known license violation.
- Qualitative:
  - User study with developers.
  - Semi structured Interviews.
- Safeguard:
  - Automated test execution to ensure quality
  - Static analysis & Compliance scanning.



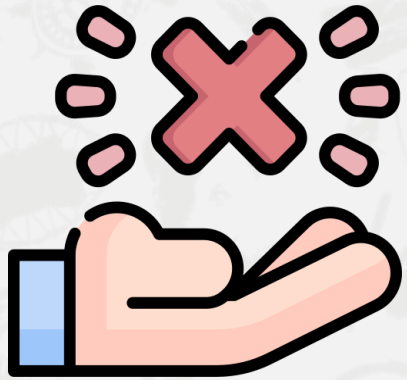
Expected Contributions

- A novel Ai-augmented framework.
- Protype tool.
- Refined metrics and models for automated code translation or generation.
- Empirical evidence on automated clone management on software quality, security and maintainability.
- Insights into developer trust.
- Guidelines for building trusthworthy, legally compliant ai assistants.



Limitations

- Language Scope.
- Performance tradeoffs.
- Ai Reliability.
- Validation Coverage.
- Study Scale.
- Scope of inclusive.



Broader Impact & Conclusion

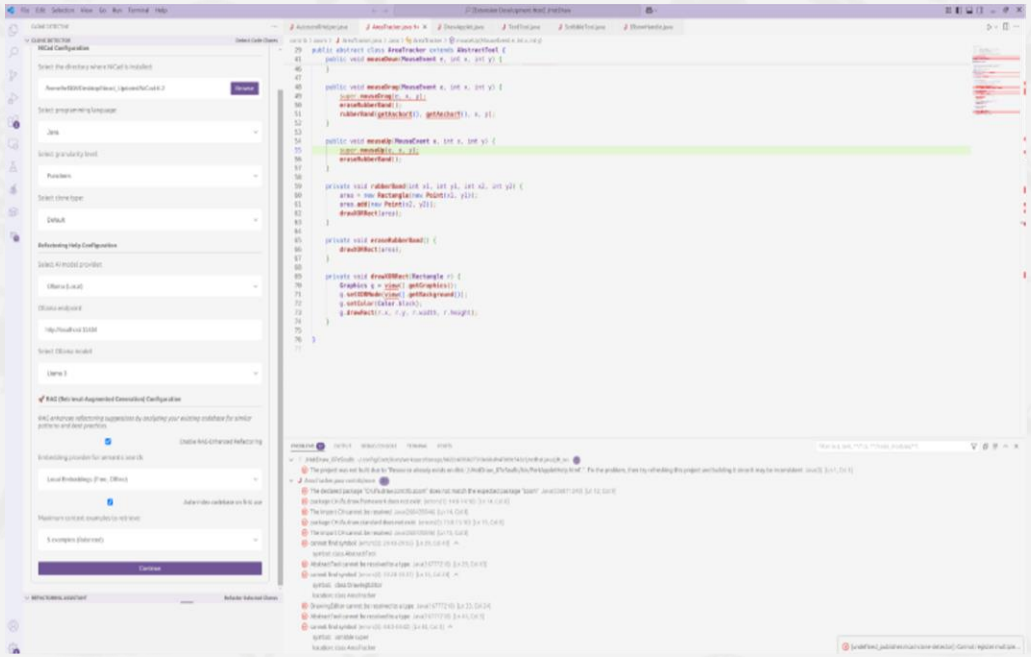
- Smarter Software Maintenance
- Enhancing developer AI collaboration
- Generalization



References

- Rahman, F., Bird, C., & Devanbu, P. (2012). Clones: What is that smell?. *Empirical Software Engineering*, *17*, 503-530.
- Wagner, S., Abdulkhaleq, A., Kaya, K., & Paar, A. (2016, March). On the relationship of inconsistent software clones and faults: An empirical study. In 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER) (Vol. 1, pp. 79-89). IEEE.
- Mondal, M., Roy, B., Roy, C. K., & Schneider, K. A. (2019). An empirical study on bug propagation through code cloning. Journal of Systems and Software, 158, 110407.
- Mandal, M., Roy, C. K., & Schneider, K. A. (2014, February). Automatic ranking of clones for refactoring through mining association rules. In 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE) (pp. 114-123). IEEE.
- Mondal, M., Roy, C. K., & Schneider, K. A. (2014, September). Automatic identification of important clones for refactoring and tracking. In 2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation (pp. 11-20). IEEE.
- Choi, E., Yoshida, N., Ishio, T., Inoue, K., & Sano, T. (2011, July). Finding code clones for refactoring with clone metrics: A case study of open source software. In Proceedings (pp. 1-5).
- Yahya, H. M., & Taha, D. B. (2023). Software Code Refactoring: A Comprehensive Review. *Journal of Education & Science*, *32*(1).
- Tsantalis, N., & Krishnan, G. P. (2013, May). Refactoring clones: A new perspective. In 2013 7th International Workshop on Software Clones (IWSC) (pp. 12-13). IEEE.
- Almatrafi, A. A., Eassa, F. A., & Sharaf, S. A. (2025). Refactoring and Clones Reduction Framework for Developing Maintainable Software Systems. International Journal of Computer Science & Network Security, 25(2), 11-22.
- Port, D., Taber, B., & Huang, L. (2023, May). Investigating a NASA cyclomatic complexity policy on maintenance risk of a critical system. In 2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 211-221). IEEE.
- Wagner, S., Abdulkhaleq, A., Kaya, K., & Paar, A. (2016, March). On the relationship of inconsistent software clones and faults: An empirical study. In 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER) (Vol. 1, pp. 79-89). IEEE.
- Wagner, S., Abdulkhaleq, A., Kaya, K., & Paar, A. (2016, March). On the relationship of inconsistent software clones and faults: An empirical study. In 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER) (Vol. 1, pp. 79-89). IEEE.
- Siegmund, J., Peitek, N., Parnin, C., Apel, S., Hofmeister, J., Kästner, C., ... & Brechmann, A. (2017, August). Measuring neural efficiency of program comprehension. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (pp. 140-150).
- Higo, Y., Ueda, Y., Kamiya, T., Kusumoto, S., & Inoue, K. (2002). On software maintenance process improvement based on code clone analysis. In Product Focused Software Process Improvement: 4th International Conference, PROFES 2002 Rovaniemi, Finland, December 9–11, 2002 Proceedings 4 (pp. 185-197). Springer Berlin Heidelberg.
- Siegmund, J., Peitek, N., Parnin, C., Apel, S., Hofmeister, J., Kästner, C., ... & Brechmann, A. (2017, August). Measuring neural efficiency of program comprehension. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (pp. 140-150).
- CHEN, H., MORIMOTO, M., SHOTA, M., & HOSOMI, T. (2025). Together We Are Better: LLM, IDE and Semantic Embedding to Assist Move Method Refactoring.
- Batole, F., Bellur, A., Dilhara, M., Ullah, M. R., Zharov, Y., Bryksin, T., ... & Dig, D. (2025). Leveraging LLMs, IDEs, and Semantic Embeddings for Automated Move Method Refactoring. arXiv preprint arXiv:2503.20934.

References



Thank You