***Andrii Tkachuk,*** *PhD student*

*NTUU “Igor Sikorsky Kyiv Polytechnic Institute”*

# Automated refactoring with semantic preservation

Automated refactoring tools provide developers the ability to improve code quality systematically and effectively. They have become an integral part of modern software development environments, helping teams maintain and evolve their code bases more efficiently.

The effective functioning of automated refactoring tools largely depends on automated analysis of the initial code. Analysis provides a necessary understanding of the code's structure and semantics, ensuring the safety and consistency of refactoring.

Rule-based refactoring provides systematic guidance to developers on improving the structure and design of the code. The key is to recognize problematic patterns (code smells) and know how to address them using established refactoring methods.

Enhancing static code analysis and refactoring mechanisms by considering the semantics of the code will provide a completer and more accurate internal model of code representation, leading to more precise and reliable refactoring results.

When analyzing a large codebase or identifying patterns within it, an ontology described in OWL and the associated logical inference capabilities can be extremely valuable.

OWL is considered superior for representing ontologies due to its higher expressiveness, compliance with Semantic Web standards, logical inference capabilities, and well-developed ecosystem of tools that allow for precise modeling and interaction. However, the choice of ontology representation language ultimately depends on specific requirements and the complexity of the domain being modeled.

Using logical inference systems (reasoners), developers can ensure that the ontologies they create are logically consistent and maximally informative, thereby maximizing their utility in applications such as knowledge discovery, data integration, and semantic search.

From the analysis of the properties and capabilities of OWL, we can conclude that the expressiveness of OWL2 profiles allows us to represent source code with all its dependencies and complexities. Constructing a model of source code based on an ontology of object-oriented programming languages using additional tools such as SWRL and SQWRL will enable the effective application of defined refactoring rules, making queries on-demand from software developers, and conducting automated analysis of source code for further improvement. Mathematically verified and justified algorithms of logical inference systems, which will be applied in the proposed model, will guarantee the consistency and coherence of refactoring actions described using various means.

Using a knowledge base that contains facts corresponding to a specific ontology is evidently an effective way to store knowledge, including knowledge about source code.

One of the main limitations of logical inference systems regarding knowledge bases built on ontologies is the inability to compute functional dependencies, which are important for describing refactoring actions and their automated execution. To expand the capabilities of classical logical inference systems, the use of SWRL rules has been proposed. SWRL rules utilize the concepts of TBox knowledge bases and allow to produce new knowledge based on functional dependencies.

It is worth mentioning that proposed refactoring method (which uses source code representation in knowledge base) preserves all semantic meanings. This means that if the source code contains drawbacks in the business logic, refactoring will preserve the drawback during the processing. The method is oriented on refactoring facilitation and general and well-known antipatterns detection and fixing. Though, it is still a developer’s responsibility to ensure that the source code does not contain functional or logical errors which are correct semantically in terms of source code syntax entities.