- **Problem**
  - Determining the detailed structural similarities and differences between source code fragments is a complex task
  - It can be applied to solve several source code analysis problems
  - As a specific application, our focus is on the study of where logging is used in the source code
  - logging is a pervasive practice and has various applications in software development and maintenance
  - It is a challenging task for developers to understand how to use logging calls in the source code
  - In this research, we would like to understand where developers log in practice, in a detailed way
- **A proposed Solution to the Problem**
  - Develop an automated approach to detect the detailed structural similarities and differences of Java classes that use logging calls
  - Our solution would:
    - classify logged Java methods into groups using a measure of similarity such that entities in each group has maximum similarity with each other and minimum similarity to other ones
    - construct a structural generalization of each group that represent the detailed structural similarities and differences of all logged Java methods in the group
- **Required Background Information**
  - Abstract Syntax Tree (AST), which is the basic structure we will use for describing software source code
  - First-order anti-unification, which is a technique used to construct the structural generalizations
  - Higher-order anti-unification modulo theories (HOAUMT), which is used to address the limitations of first-order anti-unification
  - The Jigsaw framework, an existing tool for a subset of HOAUMT, which we extend to determine potential correspondences between logged Java methods
  - Agglomerative hierarchical clustering, which is a clustering algorithm used for classifying logged Java methods into groups using a measure of similarity