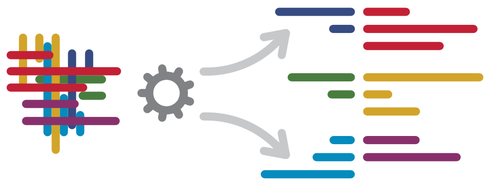
Untangling the Knot

Refactoring for Software Isolation

Summary

Refactoring software to improve its modularity enables

- capability reuse
- migration to cloud or microservices
- containerizing software
- development efficiency



Manual refactoring at scale can require months to years of development effort.[1]

Solution

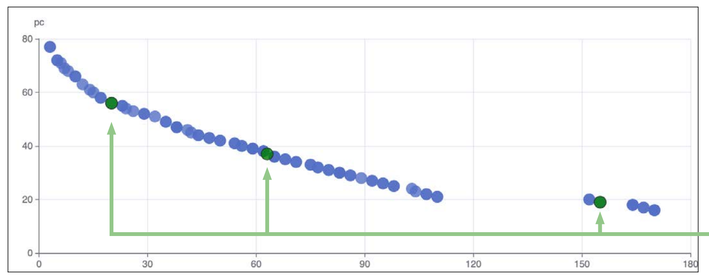
We created a refactoring assistant that analyzes existing code and generates recommendations for refactoring to improve modularity.[2] The assistant

- solves application-specific problems
- identifies common code used by multiple features or services
- provides teams with options
- works with Java, C#, and C/C++

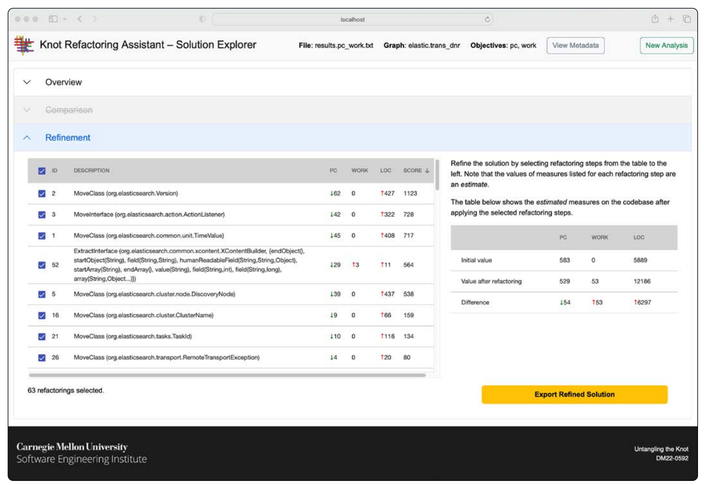
Looking Ahead

Visit **sei.cmu.edu/go/knot** for updates and contact us to discuss how the refactoring assistant could help improve the modularity of your systems.

Automated refactoring can improve the **modularity** of software at low cost.



Select refactoring recommendations that reflect different trade-offs for comparison.

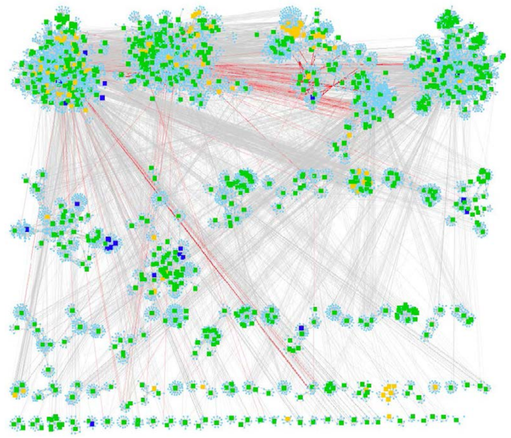


Refine refactoring recommendations to match your team's insights and context.

Each refactoring step is specific, and many can be implemented quickly using common IDE features.

The tool helps teams quickly discover which refactorings are most important to their goals.

Software structure becomes entangled over time, eroding traits like modularity.

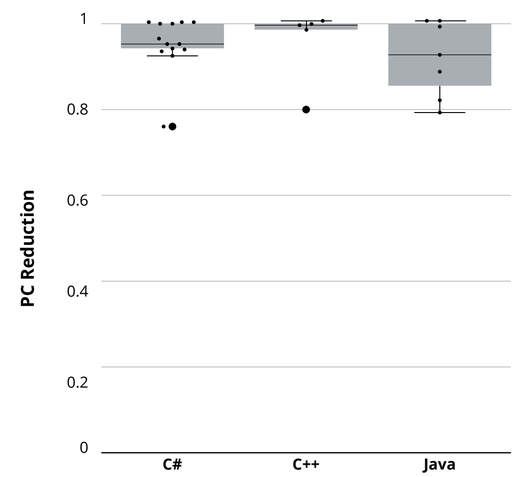


Problematic couplings

A measure of dependencies that inhibit specific modularity goals (e.g., the red lines above).

Results

Our refactoring assistant provides recommendations that reduce problematic couplings by >90% on average, significantly reducing the effort required to refactor code.



Mean overall reduction = 95.2%

1 J. Ivers, R. Nord, I. Ozkaya, C. Seifried, C. Timperley, M. Kessentini. Industry Experiences with Large-Scale Refactoring. *Foundations of Software Engineering: Software Engineering in Practice* (ESEC/FSE). November 2022.

2 J. Ivers, C. Seifried, I. Ozkaya. Untangling the Knot: Enabling Architecture Evolution with Search-Based Refactoring. *19th IEEE International Conference on Software Architecture* (ICSA 2022). March 2022.

Carnegie Mellon University
Software Engineering Institute

James Ivers | jivers@sei.cmu.edu
Chris Seifried, Jonathan Loungani, Mario Benitez, Ipek Ozkaya, Tapajit Dey, Greg Such

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM24-1446