

Untangling the Knot: Enabling Rapid Software Evolution

James Ivers

January 11, 2022

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2022 Carnegie Mellon University.

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.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM22-0021

Knot: an Automated Refactoring Assistant

The SEI has created a prototype refactoring assistant (TRL 4-5) that aims to reduce the time and effort for several common refactoring tasks.

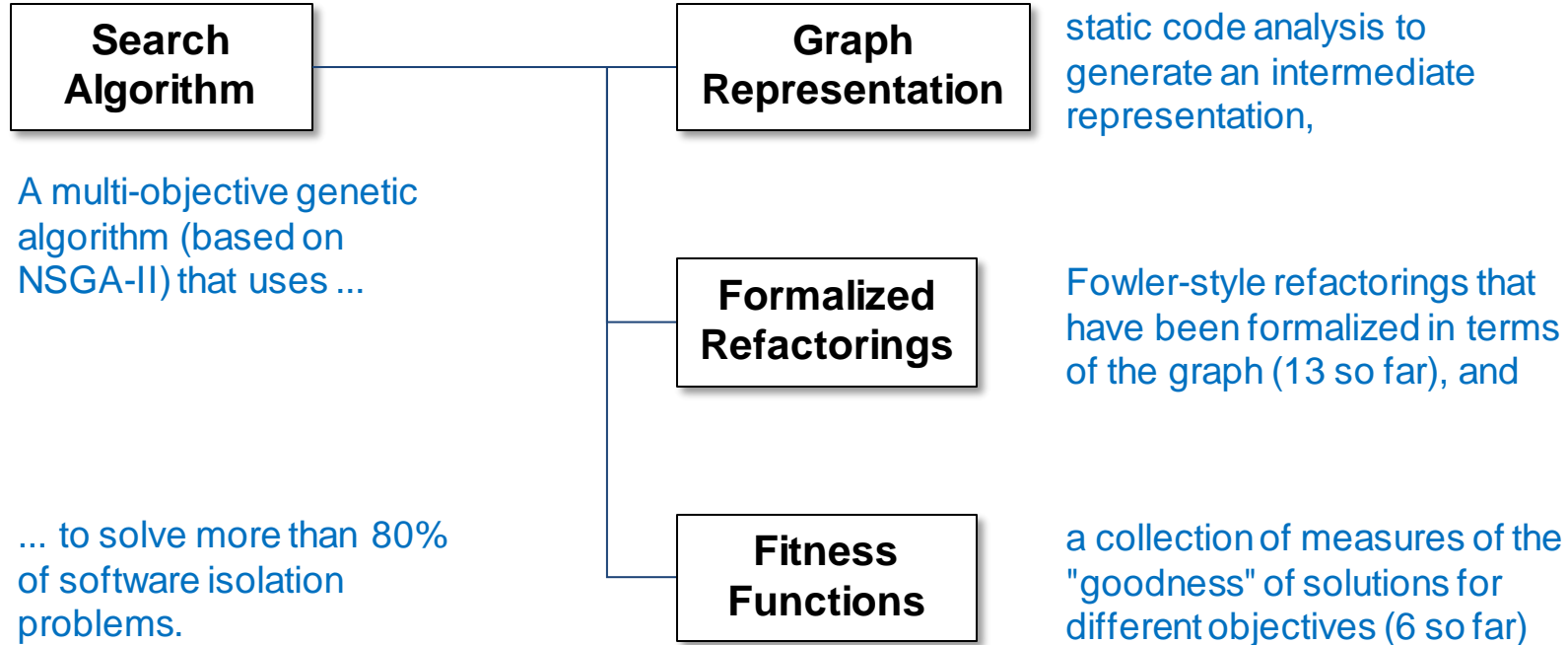
- Works on C# (now) and Java (this quarter) source code
- Scales to at least 1.2M LOC
- Reduces problematic couplings by more than 80%

Many evolution projects start with a common problem – isolating software:

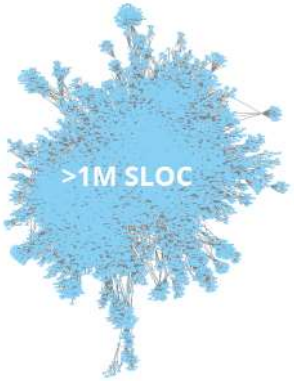
- Reusing capability in a different system or rehosting on a different platform
- Factoring out common capability as a shared asset
- Decomposing a monolith into more modular code
- Migrating capabilities to a cloud or microservice architecture

We are looking for collaborators with C# or Java code and suitable challenges.

SEI's Automated Refactoring Assistant

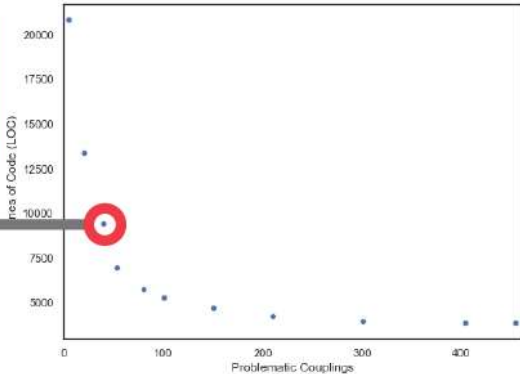
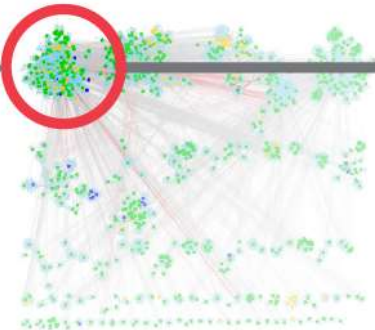


Generating Refactoring Recommendations



```
Step 1: MoveClass (Duplicati.Server.Database.Notification)
Step 2: MoveInterface (Duplicati.Server.Serialization.Interface.IBackup)
Step 3: MoveClass (Duplicati.Server.Database.Backup)
Step 4: MoveClass (Duplicati.Server.Measurement.Methods.RequestInfo)
Step 5: MoveInterface (Duplicati.Server.Serialization.Interface.ISetting)
Step 6: ExtractStaticClass (Duplicati.Library.AutoUpdater.UpdateManager,
{Duplicati.UpdateClient.MethodInfo,System.String,Duplicati.Li-
brary.AutoUpdater.AutoUpdateStrategy}, InstalledBaseDir,
INSTALLED_BASE_DIR) -> new_class_name_1
-> Supply a new meaningful name for the new class (new_class_name_1).
Step 7: MoveInstanceMethod (Duplicati.Server.EventPollNotify.SignalNewEv-
entInfo, Duplicati.Server.Database.Connection)
-> Convert the instance method to a static method by adding a new
parameter with a type of the original declaring class. Also, update all
references to this within the method to use the new parameter.
-> Convert the member Duplicati.Server.EventPollNotify_m_eventInfo to
public to allow Duplicati.Server.EventPollNotify.SignalNewEvent to
continue to access it.
-> Convert the member Duplicati.Server.EventPollNotify_m_lock to public to
allow Duplicati.Server.EventPollNotify.SignalNewEvent to continue to
access it.
-> Convert the member Duplicati.Server.EventPollNotify_m_eventInfo to
public to allow Duplicati.Server.EventPollNotify.SignalNewEvent to
continue to access it.
Step 8: MoveInterface (Duplicati.Server.Serialization.Interface.ISchedule)
Step 9: MoveClass (Duplicati.Server.Strings.Progress)
Step 10: ExtractStaticClass (Duplicati.Library.Localization.ShortLc,
{ISystem.String,System.Object}, {System.String,ISystem.String,Sys-
tem.Object,System.Object}) -> new_class_name_2
-> Supply a new meaningful name for the new class (new_class_name_2).
Step 11: ExtractStaticClass (Duplicati.Library.Common.Platform,
{IClientKindInfo,IClientPosix}) -> new_class_name_3
-> Supply a new meaningful name for the new class (new_class_name_3).
Step 12: MoveClass (Duplicati.Server.EventPollNotify)
```

Select and implement a solution that suits your context.



Select Objectives

- minimize problematic couplings
- minimize code changes
- maximize code quality
- ...

Our prototype uses a multi-objective genetic algorithm to generate a set of Pareto-optimal solutions (recommendations)