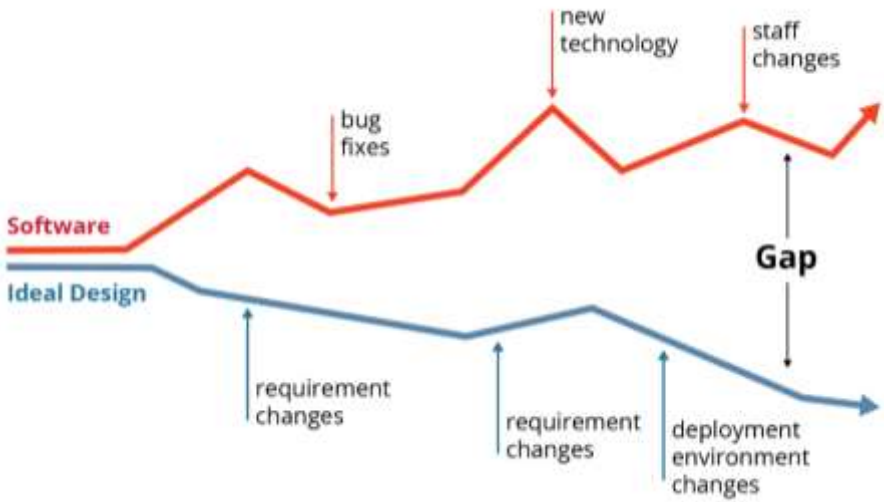
# Untangling the Knot

Mario Benitez

NOVEMBER 9, 2022

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

# Software Gets Tangled Over Time




*"Software is always changing"*

- New capabilities, bug fixes, etc., all contribute to software entropy.
- Features become dispersed across files or libraries.
- Direct dependencies to platforms or third-party libraries make it hard to change underlying technologies.
- Experts are needed to make the necessary corrections, limiting the ability for newcomers to contribute as effectively as the could.
- Refactoring software is needed to make these corrections.

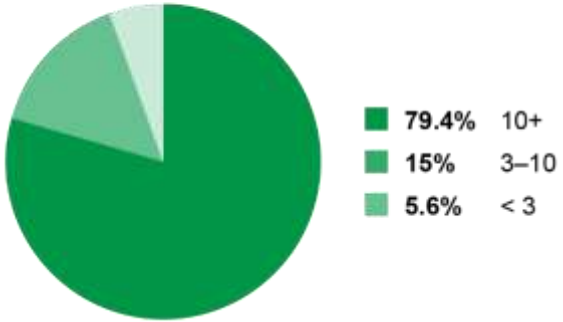
# A Survey on Large-Scale Refactoring (LSR)

We surveyed practitioners to understand how large-scale refactoring is performed today.

- How common is it?
- What motivates it?
- What makes it hard?
- What tools are used? Needed?

107 responses, 96% of whom worked as software engineers or architects.

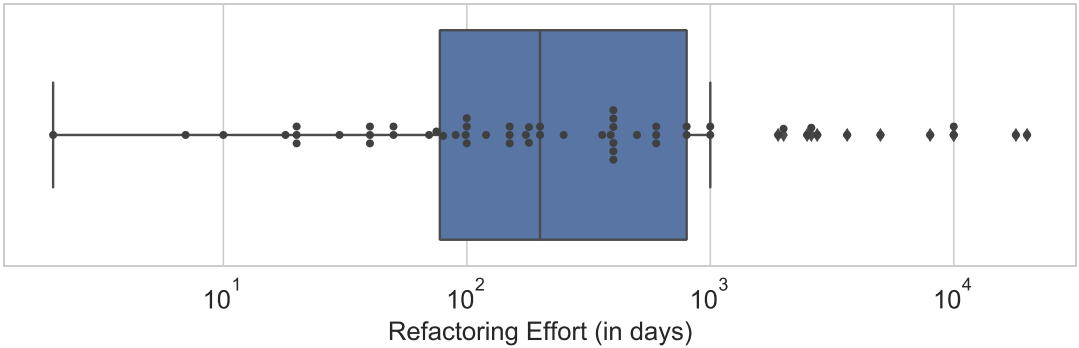
**How many years of experience did respondents have?**




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.

J. Ivers, R. Nord, I. Ozkaya, C. Seifried, C. Timperley, M. Kessentini. **Industry's Cry for Tools That Support Large-Scale Refactoring.** *Intl. Conference on Software Engineering: Software Engineering in Practice* (ICSE-SEIP). May 2022.

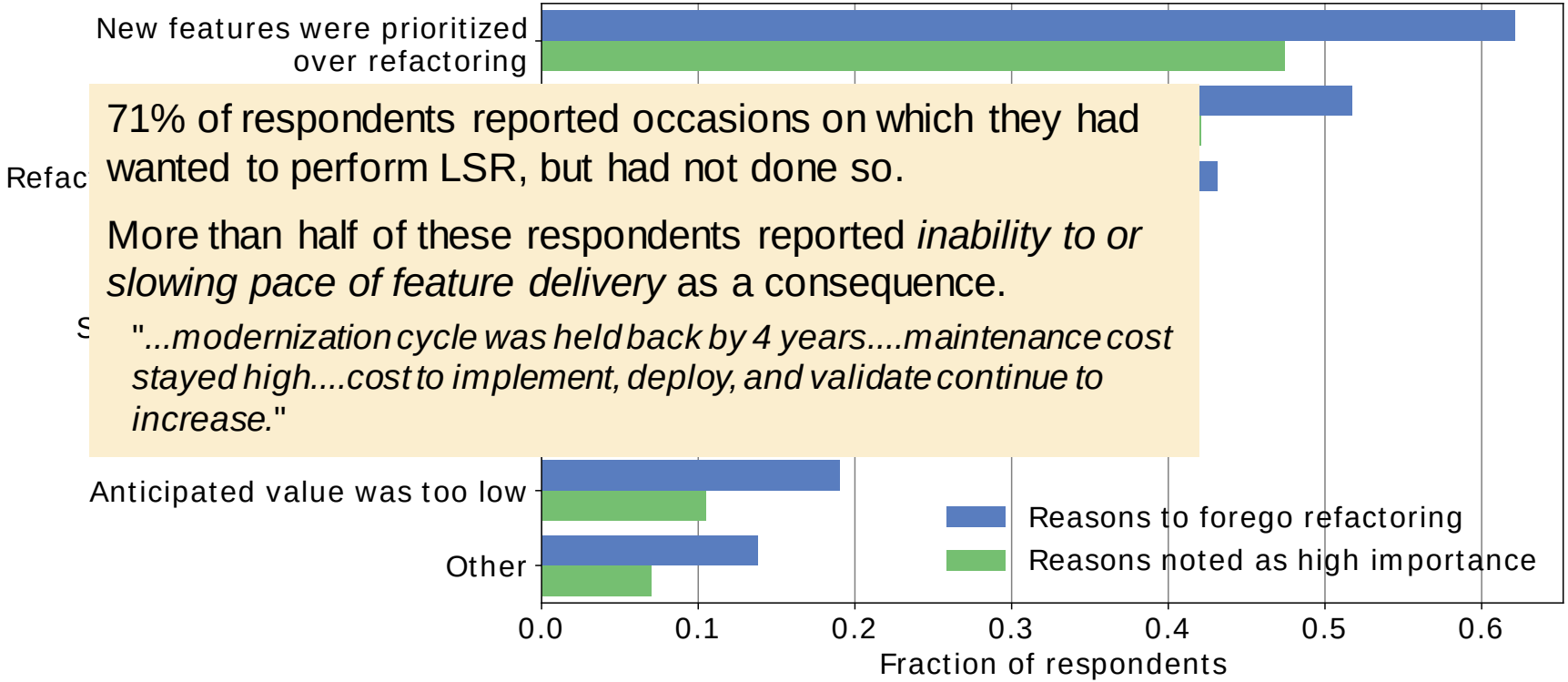
# Large-Scale Refactoring Requires Significant Effort




Mean time estimated to complete large-scale refactoring > 1,500 days

Dealing with large-scale refactoring is expensive and takes significant resources away from other priorities.

# Reasons for Forgoing LSR

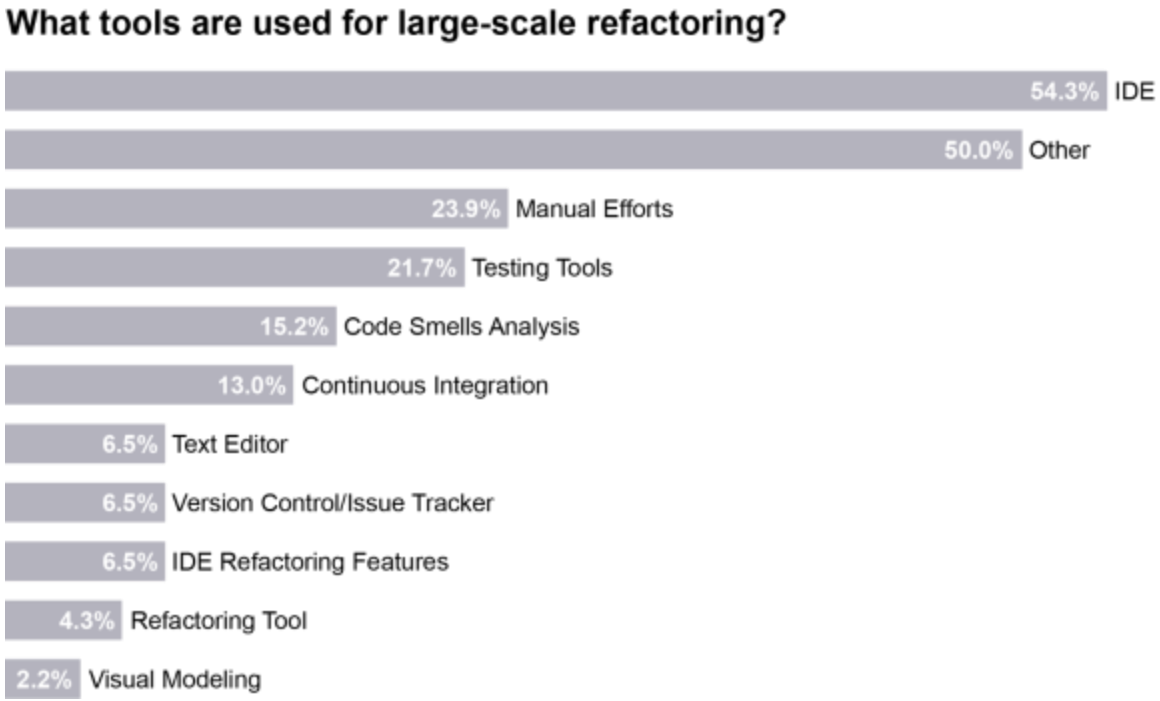



71% of respondents reported occasions on which they had wanted to perform LSR, but had not done so.

More than half of these respondents reported *inability to or slowing pace of feature delivery* as a consequence.

> "*...modernization cycle was held back by 4 years....maintenance cost stayed high....cost to implement, deploy, and validate continue to increase.*"

# Tools Used in Large-Scale Refactoring




**Refactoring tools are not widely used in LSR**

- < 10% reported using tools designed for refactoring
- Manual effort and custom scripts were reported more often than refactoring tools

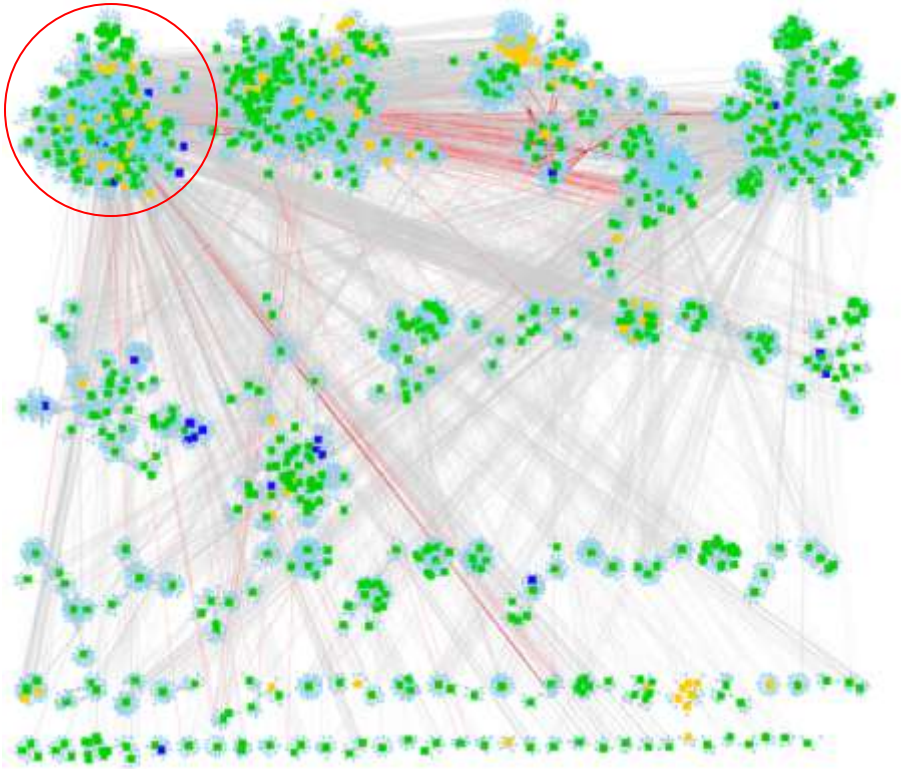
# Untangling the Knot
## Software Isolation

**Software isolation** refers to the activity of extracting specific code structures from their context for use somewhere else or with another purpose.

Examples of activities that involve software isolation:

- Containerization / Cloud Migration
- Code Reuse
- Keep code clean, understandable, and maintainable
- Replace / Abstract-away underlying technologies

# Untangling the Knot
## Key Concept – Problematic Couplings



Only certain software dependencies interfere with any particular goal.

For example, if we want to harvest a feature:

- The core problem is dependencies (red lines) from software being harvested to software that is being left behind.
- All other dependencies are irrelevant to the goal, allowing us to focus our analysis and search for solutions.

This insight enables us to apply **search-based software engineering** techniques and treat this as an **optimization problem**.

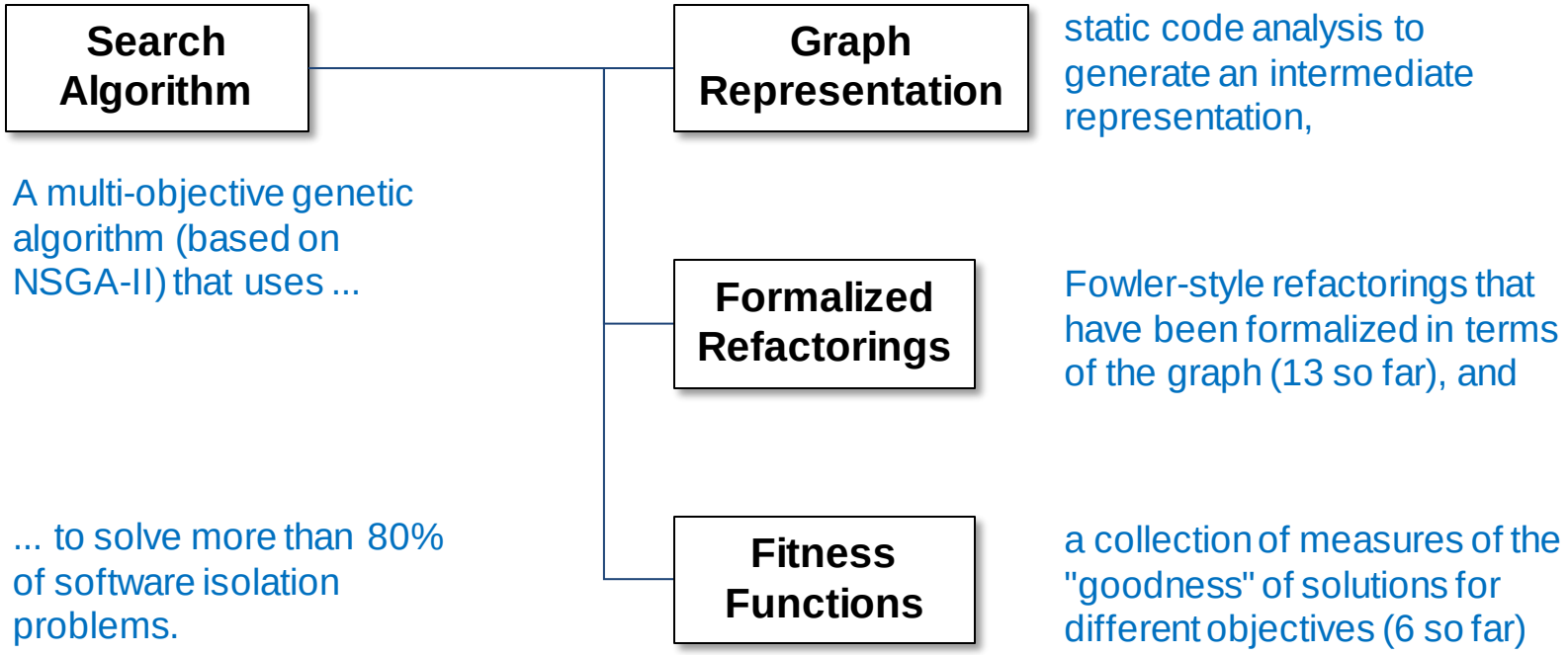
# SEI's Automated Refactoring Assistant

We have developed an automated refactoring assistant for developers that improves software structure for several common forms of change that involve software isolation:

- Solves project-specific problems
- Uses a semi-automated approach
- Our goal is to complete refactorings for software isolation in less than 1/3 of the time required by manual approaches


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).* 2022.

# SEI's Automated Refactoring Assistant



# Automated Refactoring Assistant
## Example Project-specific Goal

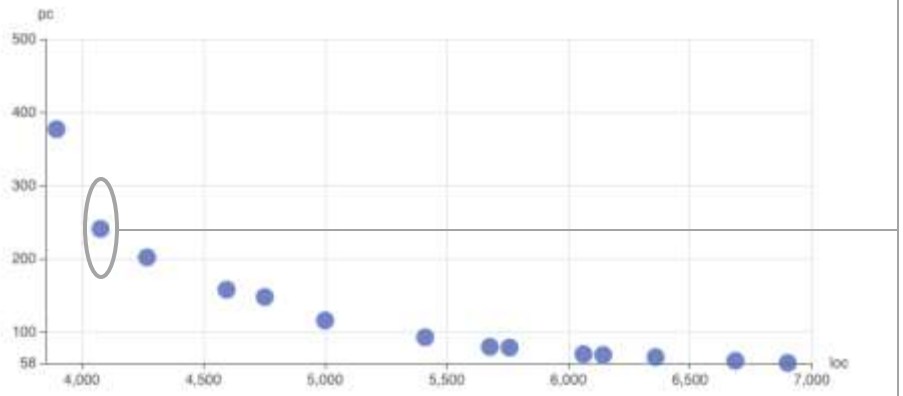
**example.set**

```
Duplicati.Server.Database.Connection
Duplicati.Server.Database.ServerSettings
Duplicati.Server.Program
Duplicati.Server.Serializable.ImportExportStructure
Duplicati.Server.Serialization.CloseReason
Duplicati.Server.Serialization.DuplicatiOperation
Duplicati.Server.Serialization.Interface.INotification
Duplicati.Server.Serialization.Interface.IServerStatus
Duplicati.Server.Serialization.LiveControlState
Duplicati.Server.Serialization.NotificationType
Duplicati.Server.Serialization.RunnerResult
Duplicati.Server.Serialization.RunnerState
Duplicati.Server.Serialization.Serializer
Duplicati.Server.Serialization.SuggestedStatusIcon
Duplicati.Server.Serialization.UpdatePollerStates
Duplicati.Server.SingleInstance
Duplicati.Server.WebServer.RESTMethods.Backups
Duplicati.Server.WebServer.Server
```

*List of types users wish to extract using the refactoring assistant*

- Users identify which capabilities (classes, interfaces, methods) need to be extracted to accomplish the project goals.
- Once the capabilities have been identified they can be easily be specified in a simple text file by hand. No tools necessary.

# Multi-objective Optimization

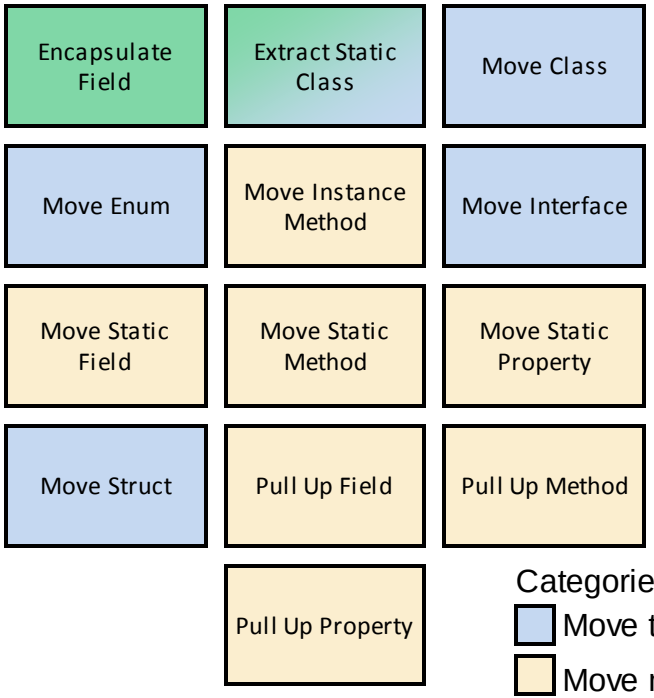



Our refactoring assistant generates a collection of Pareto-optimal solutions that represent trade-offs among competing objectives.

▾ Solution 12 -- aec = 4, dec = 0, loc = 4077, lscc = 0.56, pc = 241, rfLen = 18, rfOptions = 95, sparsity = 50, typesChanged = 8

- Step 1: ExtractStaticClass (Duplicati.Library.Utility.Utility, {ForceStreamRead(Stream,byte[],int), ParseBoolOption(IDictionary<string,string>,string), ClientFilenameStringComparison, ParseBool(string,bool), ReadFileWithDefaultEncoding(string), EPOCH}) -> new_class_name_1
- Step 2: ExtractStaticClass (Duplicati.Library.Utility.Timeparser, {ParseTimeSpan(string), ParseTimeInterval(string,DateTime)}) -> new_class_name_2
- Step 3: ExtractStaticClass (Duplicati.Library.AutoUpdater.UpdaterManager, {RunFromMostRecent(System.Reflection.MethodInfo,string[],AutoUpdateStrategy), InstalledBaseDir, INSTALLED_BASE_DIR}) -> new_class_name_3
- Step 4: ExtractStaticClass (Duplicati.Library.Common.Platform, {IsClientWindows, IsClientPosix}) -> new_class_name_4
- Step 5: ExtractStaticClass (Duplicati.Library.Utility.Utility, {IsFSCaseSensitive, CachedIsFSCaseSensitive}) -> new_class_name_5
- Step 6: MoveInterface (Duplicati.Server.Serialization.Interface.ISetting)
- Step 7: MoveInterface (Duplicati.Server.Serialization.Interface.IBackup)
- Step 8: MoveClass (Duplicati.Server.Strings.Program)
- Step 9: ExtractStaticClass (Duplicati.Library.Localization.Short.LC, {L(string,object), L(string), L(string,object,object)}) -> new_class_name_6
- Step 10: MoveInterface (Duplicati.Server.Serialization.Interface.ISchedule)
- Step 11: MoveClass (Duplicati.Server.Database.TempFile)
- Step 12: MoveClass (Duplicati.Server.Database.Notification)
- Step 13: MoveInstanceMethod (Duplicati.Server.WebServer.RESTMethods.RequestInfo.ReportClientError(string,System.Net.HttpStatusCode), Duplicati.Server.WebServer.RESTMethods.Backups)
- Step 14: MoveInstanceMethod (Duplicati.Server.EventPollNotify.SignalNewEvent(), Duplicati.Server.Database.Connection)
- Step 15: MoveClass (Duplicati.Server.Strings.Server)
- Step 16: MoveClass (Duplicati.Library.AutoUpdater.UpdateInfo)
- Step 17: MoveEnum (Duplicati.Library.UsageReporter.ReportType)
- Step 18: MoveInterface (Duplicati.Server.Serialization.Interface.IFilter)

# Automated Refactoring Assistant
## Refactoring Operations




- Refactoring operations define the steps to be taken when isolating the desired capability.
- Many refactorings are familiar from Fowler's catalog.
- Some (Move Class, Move Enum, and Move Interface) are novel because we have an additional concept where a new unit or project with the initial list of classes (defined in the candidate set) implementing the capability of interest is created to begin the refactoring process.

# Our Vision

Architects live in brown-field development and need tools that help them return their code to a healthier state.

Traits that we are pursuing include

- Focusing on industry-driven problems
- Leveraging explicit user intent
- Providing transparency
- Embracing an assistant model and partial solutions

**<u>This is where we are today:</u>**

- Support for two different programing languages: C# and Java
- Scales to at least 1.2M SLOC
- Generates recommendations that solve most of each software isolation problem (87.9% reduction for C# and 69.3% reduction for Java)
- Tool can perform analysis in a few hours.

# THANK YOU!

For more information visit our website:

- https://www.sei.cmu.edu/go/knot

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.



DM22-1035