Untangling the Knot: Recommending Refactorings

James Ivers

May 7, 2020

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

[Distribution Statement A] Approved for public release and unlimited distribution.

Copyright 2020 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.

DM20-0380

Topics

Project Overview

Near-term Potential

Long-term Potential

Wrap Up

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

CMU SEI is a DoD Federally Funded Research and Development Center

Our mission: Engineering and securing software

Established in 1984 at Carnegie Mellon University

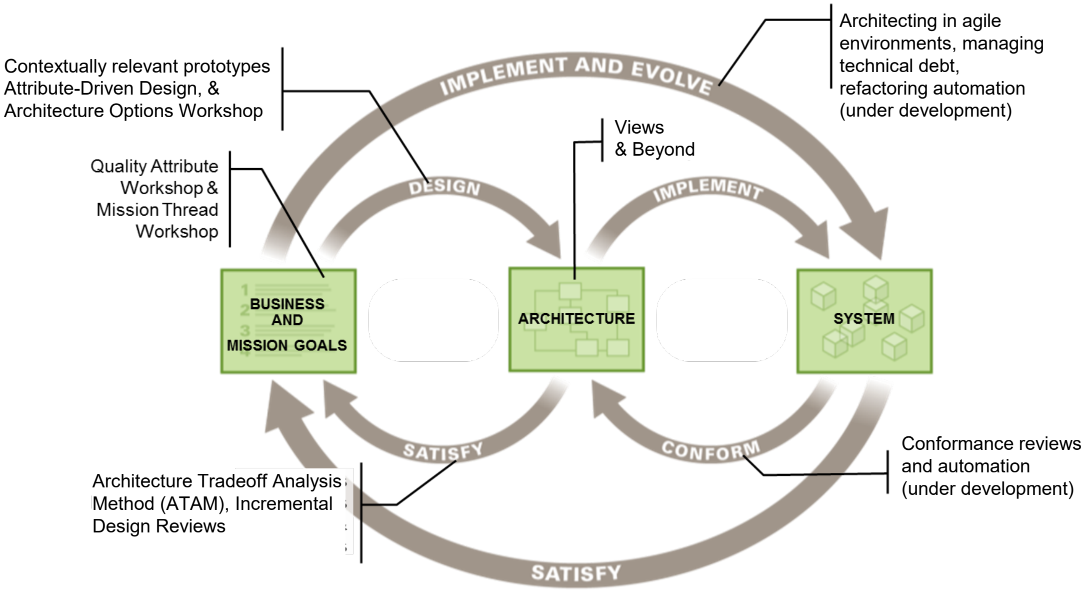
~635 employees

Offices in Pittsburgh and DC, with locations near customer facilities in MA, MD, TX, and CA

~$145M in annual funding (~$20M USD(R&E) 6.2 and 6.3 Line funding)

[Distribution Statement A] Approved for public release and unlimited distribution.

Establishing a Discipline of Software Architecture



Contextually relevant prototypes Attribute-Driven Design, & Architecture Options Workshop

Quality Attribute Workshop & Mission Thread Workshop

Views & Beyond

Architecting in agile environments, managing technical debt, refactoring automation (under development)

Architecture Tradeoff Analysis Method (ATAM), Incremental Design Reviews

Conformance reviews and automation (under development)

Range of methods and practices applicable at different points in the development lifecycle.

- Domains of expertise include IT, C2, tactical, and health informatics
- Technology expertise includes IoT, big data, digital twin, cloud, and machine learning

10+ courses, available in a mix of public, on-site, and eLearning options.

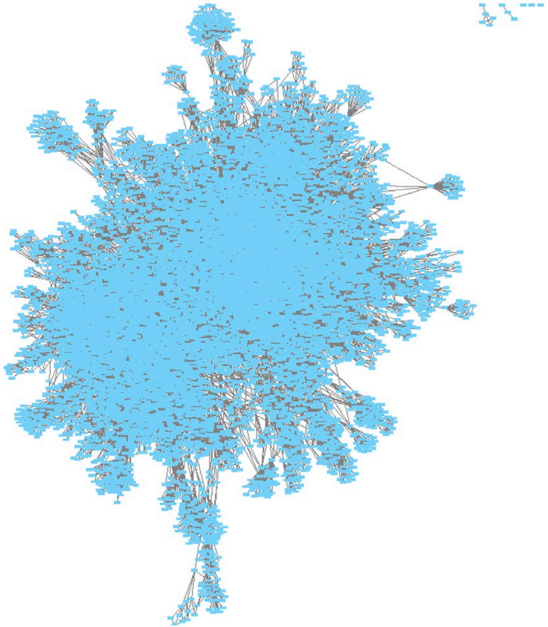
3 professional certificates.

A collection of books for wide dissemination.

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Project Overview



© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Software Structure Enables Our Ability to Innovate

Quickly delivering new capabilities and taking advantage of new technology depend on an ability to evolve software efficiently. The structure of legacy software, however, often fails to support this goal.

A recent anecdote from a DoD contractor: The estimate for isolating a mission capability from the underlying hardware platform was 14,000 staff hours (development only).

This is representative of a class of changes that involve *feature isolation* – isolating a specific software capability from its context.

Other examples include

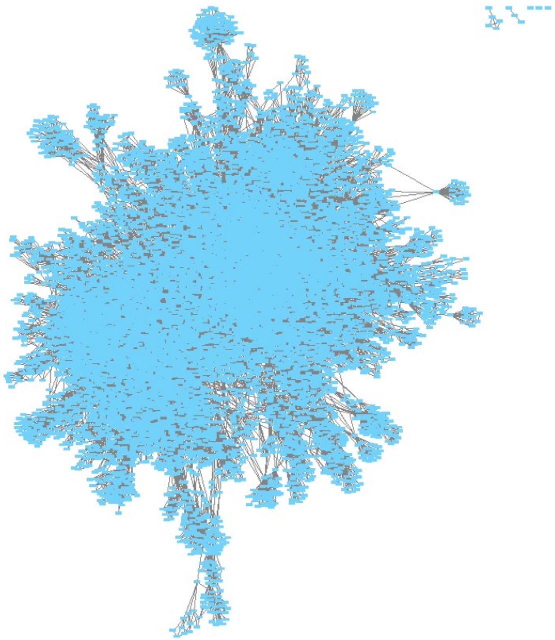
- migrating a capability to the cloud
- harvesting a component for reuse
- replacing a proprietary component

Our project aims to allow feature isolation to be done in **one-third** of the time.

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Software Complexity Is a Driver of the Effort Required



Even modest systems are hard to comprehend, and harder to modify.

- A modest application with only 68K lines of code (LOC) contains more than 10K nodes and 50K relations.
- Making a "simple" change, like isolating the code for deployment as a service, can require reasoning about hundreds of dependencies.

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Our Goal: Create an Automated Refactoring Assistant

Refactoring is a technique for improving the structure of software, but it is typically a *labor-intensive* process in which developers must

- figure out where changes are needed
- figure out which refactoring(s) to use
- implement refactorings by rewriting code

Our goal is to create an automated assistant for developers that recommends refactorings to isolate software, allowing features to be harvested or replaced in 1/3 of the time it takes to do so manually.

- Uses a semi-automated approach
- Addresses all three labor-intensive activities

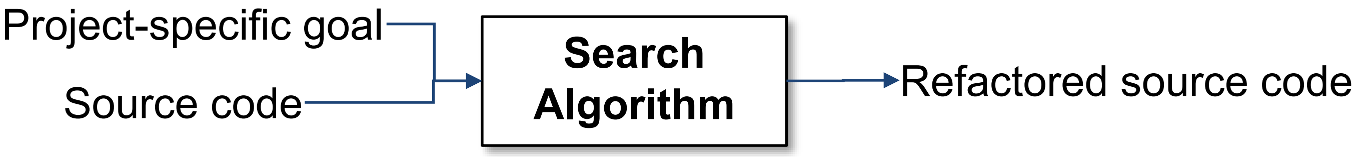
In perspective, our work would reduce the cost in the earlier example from 14,000 staff hours to 4,500 staff hours—saving the cost of 9,500 hours of development.

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Our Approach

We are adapting search-based optimization algorithms to recommend refactorings that isolate software to support harvesting or replacing features.



© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Feature Isolation Problems

Many software changes become much easier after isolating a feature from its context. We are focusing on two primary use cases: harvesting and replacing.

Harvesting software involves moving a feature from one context to another

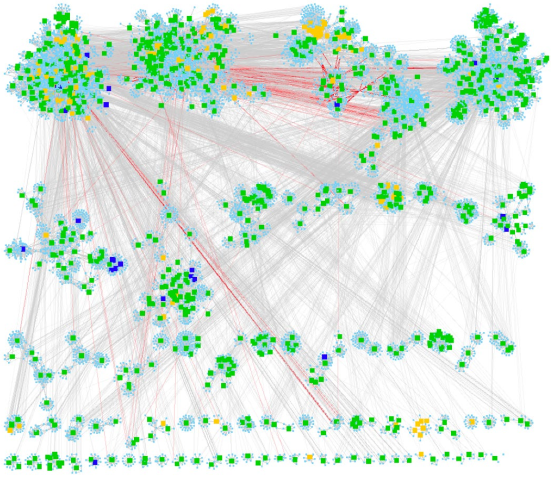
- Reusing capability in a different system or rehosting on a different platform
- Factoring out common capability as a shared asset
- Decomposing a monolith into more modular code
- Migrating services to a cloud or microservice architecture

Replacing software involve removing a feature in favor of another option

- Better options from another supplier
- Removing proprietary/licensed code

Near-term Potential

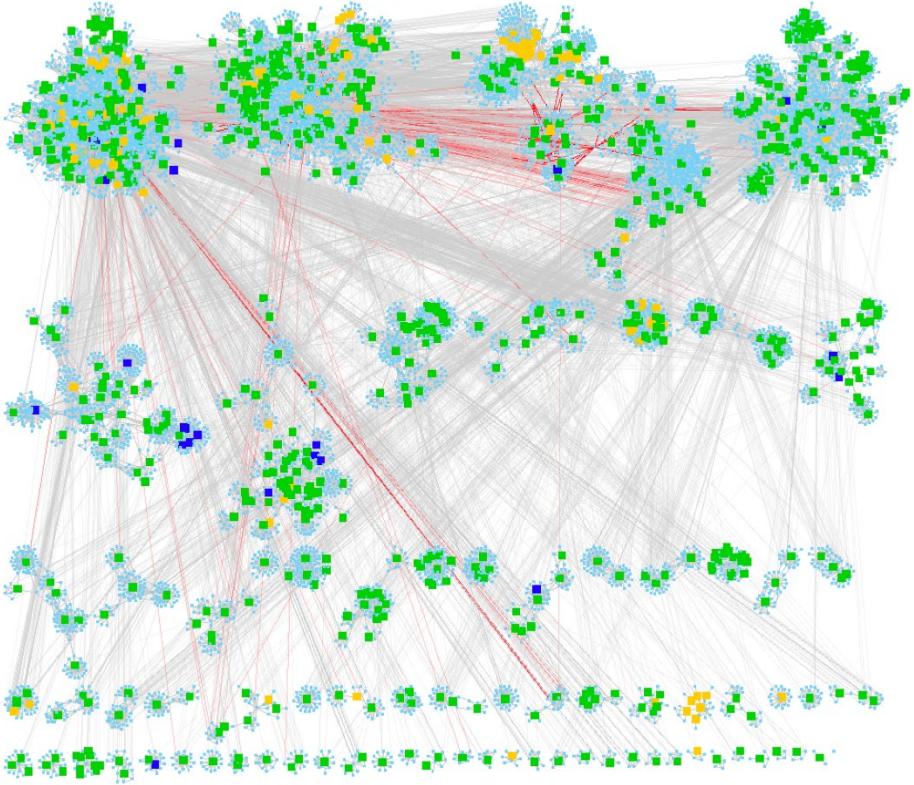
(i.e., we can do this now)



© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Problematic Couplings



Basis: Only certain software dependencies interfere with the goal.

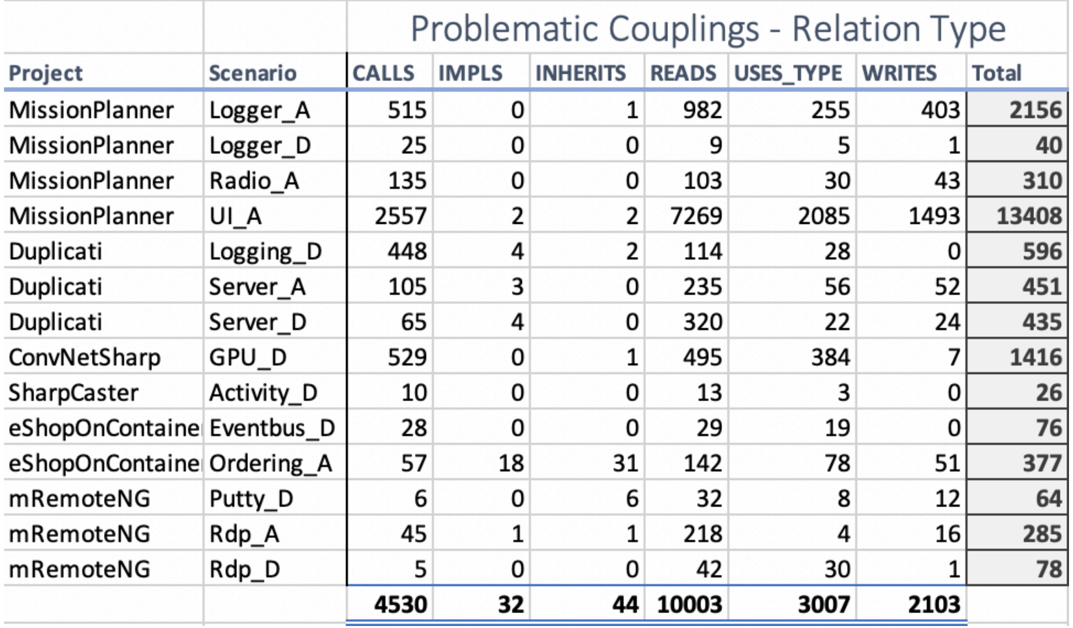
If attempting to harvest a feature

- dependencies from software moving to a new context to software that isn't moving are the core problem (red lines)
- all other dependencies are irrelevant to the task, allowing us to focus our analysis and search for solutions

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Analysis of Problematic Couplings



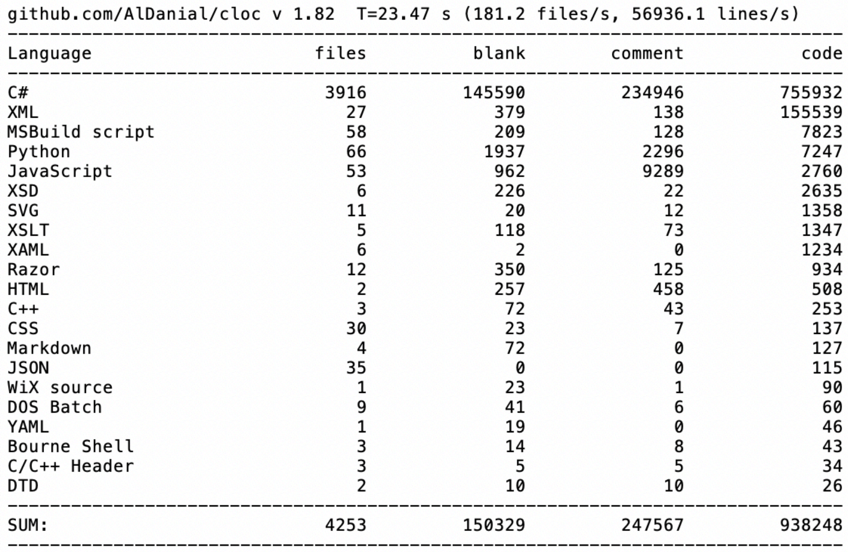
Project	Scenario	Problematic Couplings - Relation Type						
		CALLS	IMPLS	INHERITS	READS	USES_TYPE	WRITES	Total
MissionPlanner	Logger_A	515	0	1	982	255	403	2156
MissionPlanner	Logger_D	25	0	0	9	5	1	40
MissionPlanner	Radio_A	135	0	0	103	30	43	310
MissionPlanner	UI_A	2557	2	2	7269	2085	1493	13408
Duplicati	Logging_D	448	4	2	114	28	0	596
Duplicati	Server_A	105	3	0	235	56	52	451
Duplicati	Server_D	65	4	0	320	22	24	435
ConvNetSharp	GPU_D	529	0	1	495	384	7	1416
SharpCaster	Activity_D	10	0	0	13	3	0	26
eShopOnContaine	Eventbus_D	28	0	0	29	19	0	76
eShopOnContaine	Ordering_A	57	18	31	142	78	51	377
mRemoteNG	Putty_D	6	0	6	32	8	12	64
mRemoteNG	Rdp_A	45	1	1	218	4	16	285
mRemoteNG	Rdp_D	5	0	0	42	30	1	78
		4530	**32**	**44**	**10003**	**3007**	**2103**	

- Code size spans 6K to 750K source LOC
- Scenarios illustrate a range of difficulty – 26 to 13K problematic couplings
- This information can be further analyzed to understand the complexity of a proposed change.

An Illustration of this Analysis

We've analyzed a number of open source projects and one commercial project.

The motivating scenario for this analysis is a desire to **replace the existing pub/sub mechanism** with a better option.



```
github.com/AlDanial/cloc v 1.82  T=23.47 s (181.2 files/s, 56936.1 lines/s)
--------------------------------------------------------------------------------
Language                      files          blank        comment           code
--------------------------------------------------------------------------------
C#                             3916         145590         234946         755932
XML                              27            379            138         155539
MSBuild script                   58            209            128           7823
Python                           66           1937           2296           7247
JavaScript                       53            962           9289           2760
XSD                               6            226             22           2635
SVG                              11             20             12           1358
XSLT                              5            118             73           1347
XAML                              6              2              0           1234
Razor                            12            350            125            934
HTML                              2            257            458            508
C++                               3             72             43            253
CSS                              30             23              7            137
Markdown                          4             72              0            127
JSON                             35              0              0            115
WiX source                        1             23              1             90
DOS Batch                         9             41              6             60
YAML                              1             19              0             46
Bourne Shell                      3             14              8             43
C/C++ Header                      3              5              5             34
DTD                               2             10             10             26
--------------------------------------------------------------------------------
SUM:                           4253         150329         247567         938248
--------------------------------------------------------------------------------
```

Data is from an open source project (https://github.com/duplicati/duplicati).

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Graph Data

NODES [10207]	
Namespaces =	79
Classes =	866
Interfaces =	132
Structs =	24
Fields =	2094
Properties =	3160
Methods =	3278
Delegates =	13
Events =	21
Enums =	80
Files =	460

RELATIONSHIPS	[49696]
Calls =	7986
Reads =	15448
Writes =	3984
Inherits =	179
Implements =	267
Locations =	9670
Declares =	9656
Type Uses =	2506

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Problematic Couplings for the Scenario

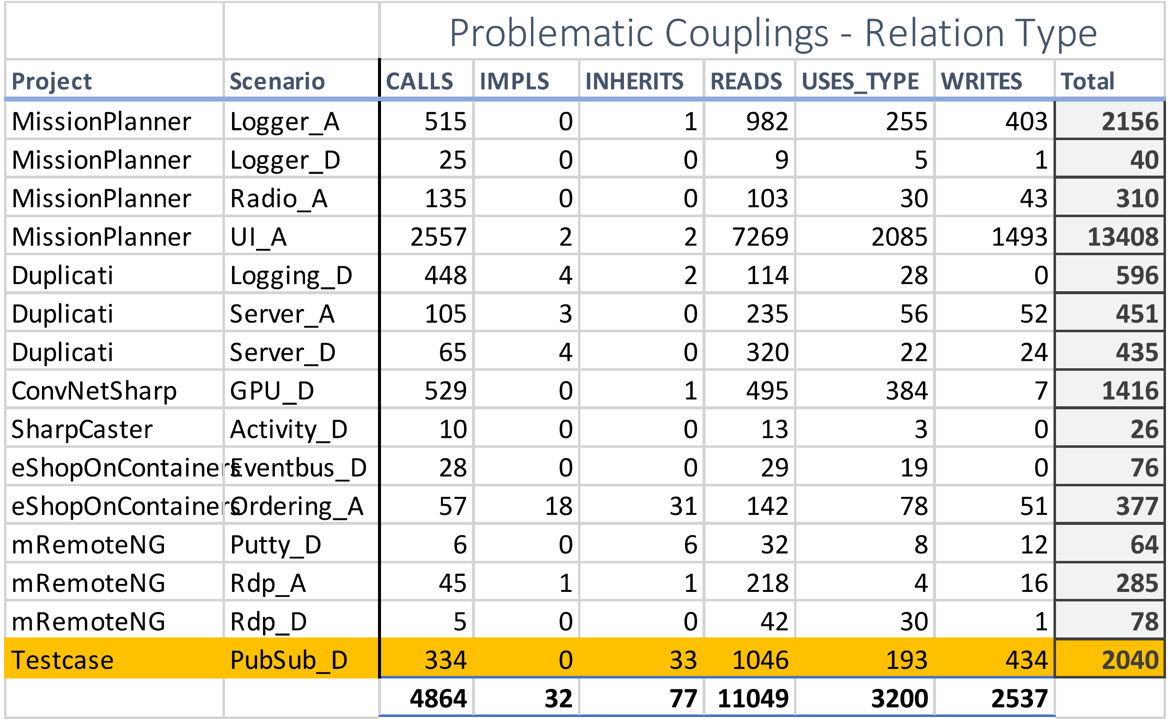
Initial problematic coupling count: **2,040**

- These are dependencies from code not being replaced to code that is being replaced
- Each occurrence is counted separately (e.g., two dependencies on the same method will be counted individually)

What does this mean?

We can slice the data a few different ways...

What Kinds of Dependencies did We Find?



		Problematic Couplings - Relation Type						
Project	**Scenario**	**CALLS**	**IMPLS**	**INHERITS**	**READS**	**USES_TYPE**	**WRITES**	**Total**
MissionPlanner	Logger_A	515	0	1	982	255	403	**2156**
MissionPlanner	Logger_D	25	0	0	9	5	1	**40**
MissionPlanner	Radio_A	135	0	0	103	30	43	**310**
MissionPlanner	UI_A	2557	2	2	7269	2085	1493	**13408**
Duplicati	Logging_D	448	4	2	114	28	0	**596**
Duplicati	Server_A	105	3	0	235	56	52	**451**
Duplicati	Server_D	65	4	0	320	22	24	**435**
ConvNetSharp	GPU_D	529	0	1	495	384	7	**1416**
SharpCaster	Activity_D	10	0	0	13	3	0	**26**
eShopOnContainers	Eventbus_D	28	0	0	29	19	0	**76**
eShopOnContainers	Ordering_A	57	18	31	142	78	51	**377**
mRemoteNG	Putty_D	6	0	6	32	8	12	**64**
mRemoteNG	Rdp_A	45	1	1	218	4	16	**285**
mRemoteNG	Rdp_D	5	0	0	42	30	1	**78**
Testcase	PubSub_D	334	0	33	1046	193	434	**2040**
		4864	**32**	**77**	**11049**	**3200**	**2537**	

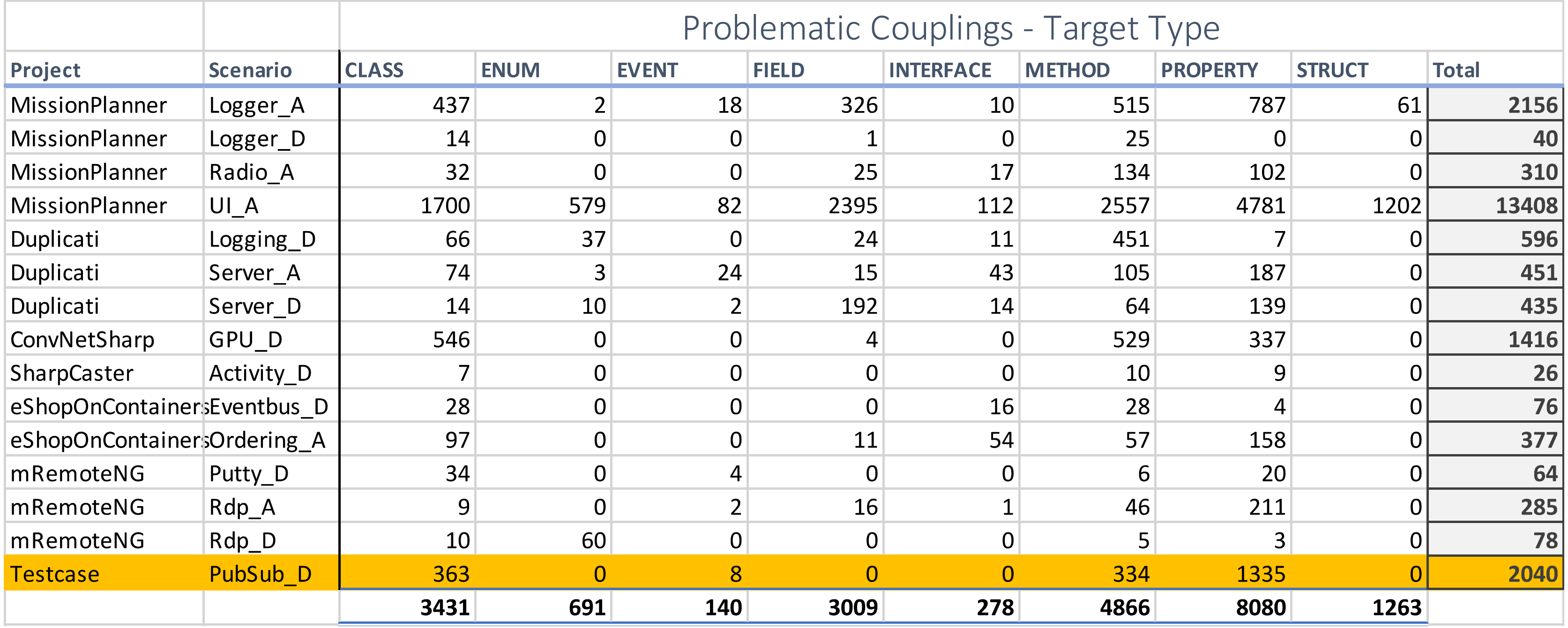
Data looks pretty similar to open source examples that we're studying

- Most common relations involve reading/writing fields
- Method calls are common
- Inheritance and and interface implementation aren't common, but are more than in other cases

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

What do the Dependencies Point To? - 1



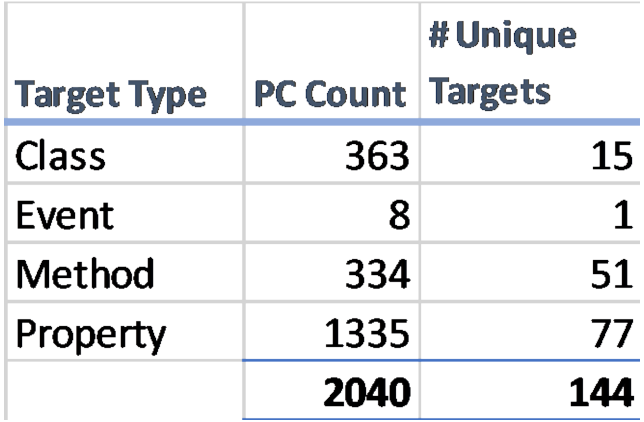
		Problematic Couplings - Target Type								
Project	Scenario	CLASS	ENUM	EVENT	FIELD	INTERFACE	METHOD	PROPERTY	STRUCT	Total
MissionPlanner	Logger_A	437	2	18	326	10	515	787	61	2156
MissionPlanner	Logger_D	14	0	0	1	0	25	0	0	40
MissionPlanner	Radio_A	32	0	0	25	17	134	102	0	310
MissionPlanner	UI_A	1700	579	82	2395	112	2557	4781	1202	13408
Duplicati	Logging_D	66	37	0	24	11	451	7	0	596
Duplicati	Server_A	74	3	24	15	43	105	187	0	451
Duplicati	Server_D	14	10	2	192	14	64	139	0	435
ConvNetSharp	GPU_D	546	0	0	4	0	529	337	0	1416
SharpCaster	Activity_D	7	0	0	0	0	10	9	0	26
eShopOnContainers	Eventbus_D	28	0	0	0	16	28	4	0	76
eShopOnContainers	Ordering_A	97	0	0	11	54	57	158	0	377
mRemoteNG	Putty_D	34	0	4	0	0	6	20	0	64
mRemoteNG	Rdp_A	9	0	2	16	1	46	211	0	285
mRemoteNG	Rdp_D	10	60	0	0	0	5	3	0	78
Testcase	PubSub_D	363	0	8	0	0	334	1335	0	2040
		3431	**691**	**140**	**3009**	**278**	**4866**	**8080**	**1263**	

By type, most dependencies are on Properties, Classes, and Methods.

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

What do the Dependencies Point To? - 2



Target Type	PC Count	# Unique Targets
Class	363	15
Event	8	1
Method	334	51
Property	1335	77
	2040	**144**

By name, dependencies concentrate on a smaller number of elements—144 unique names that collectively cover 2040 dependencies.

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Where are All the Types Defined?

All 15 classes are defined in namespaces

- The classes are declared across 11 namespaces
- There are no dependencies on nested classes

All 129 elements (Events, Methods, and Properties) are all defined in classes

- Specifically, they are defined across 17 classes
- 9 of these classes are not included in the above list of 15 classes

So, basically there are dependencies on 24 classes (or members thereof)

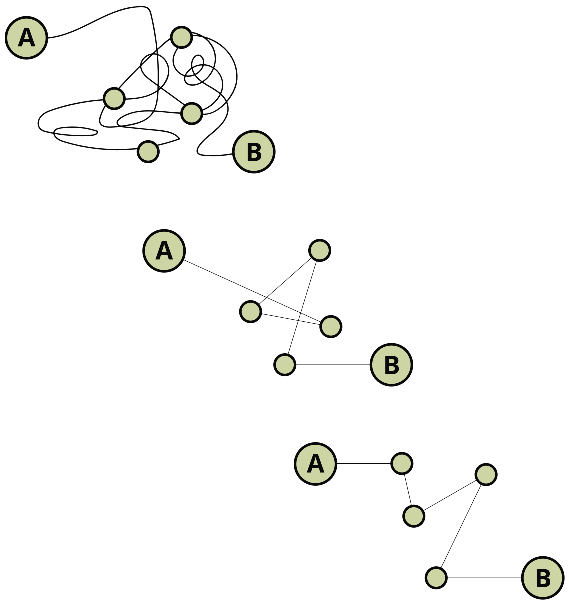
- The initial scenario identified 53 classes, of which less than half are issues in a replacement scenario

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Long-term Potential

(i.e., give us 6-12 months)

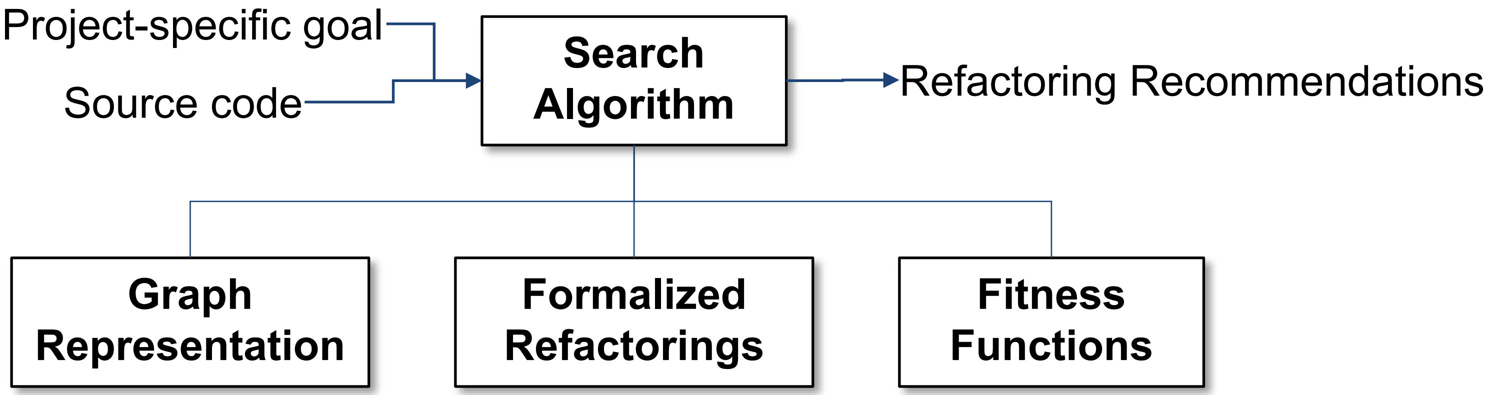


© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

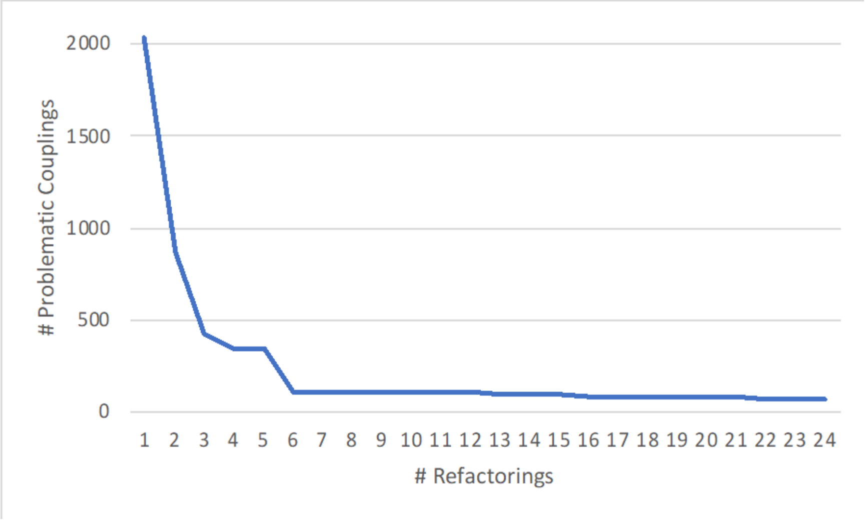
Our Approach

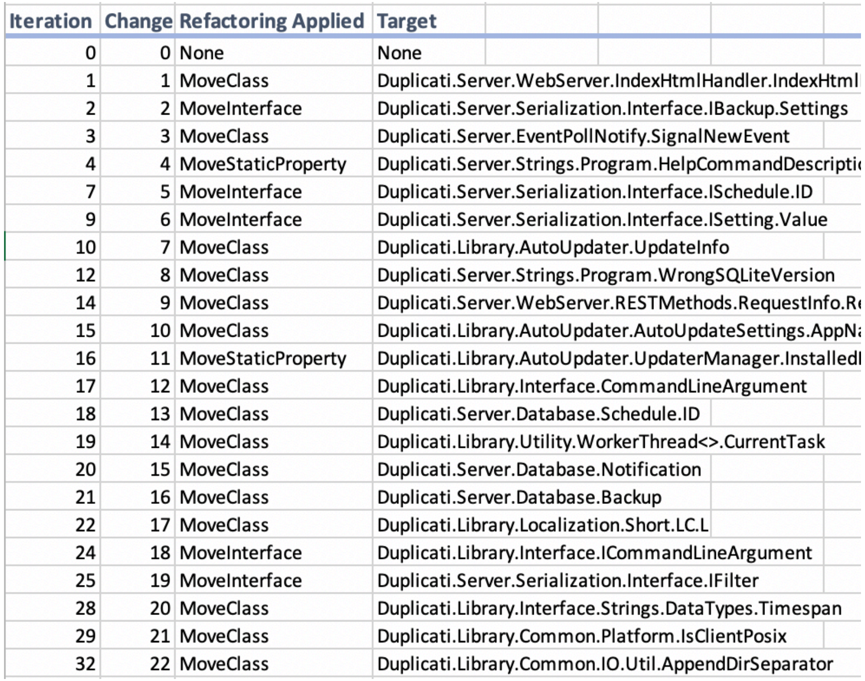
We are adapting search-based optimization algorithms to recommend refactorings that isolate software to support harvesting or replacing capabilities.



© 2019 Carnegie Mellon University
[Distribution Statement A] Approved for public release and unlimited distribution.

Initial (Local) Search Results





Iteration	Change	Refactoring Applied	Target
0	0	None	None
1	1	MoveClass	Duplicati.Server.WebServer.IndexHtmlHandler.IndexHtml
2	2	MoveInterface	Duplicati.Server.Serialization.Interface.IBackup.Settings
3	3	MoveClass	Duplicati.Server.EventPollNotify.SignalNewEvent
4	4	MoveStaticProperty	Duplicati.Server.Strings.Program.HelpCommandDescripti
7	5	MoveInterface	Duplicati.Server.Serialization.Interface.ISchedule.ID
9	6	MoveInterface	Duplicati.Server.Serialization.Interface.ISetting.Value
10	7	MoveClass	Duplicati.Library.AutoUpdater.UpdateInfo
12	8	MoveClass	Duplicati.Server.Strings.Program.WrongSQLiteVersion
14	9	MoveClass	Duplicati.Server.WebServer.RESTMethods.RequestInfo.R
15	10	MoveClass	Duplicati.Library.AutoUpdater.AutoUpdateSettings.AppN
16	11	MoveStaticProperty	Duplicati.Library.AutoUpdater.UpdaterManager.Installed
17	12	MoveClass	Duplicati.Library.Interface.CommandLineArgument
18	13	MoveClass	Duplicati.Server.Database.Schedule.ID
19	14	MoveClass	Duplicati.Library.Utility.WorkerThread<>.CurrentTask
20	15	MoveClass	Duplicati.Server.Database.Notification
21	16	MoveClass	Duplicati.Server.Database.Backup
22	17	MoveClass	Duplicati.Library.Localization.Short.LC.L
24	18	MoveInterface	Duplicati.Library.Interface.ICommandLineArgument
25	19	MoveInterface	Duplicati.Server.Serialization.Interface.IFilter
28	20	MoveClass	Duplicati.Library.Interface.Strings.DataTypes.Timespan
29	21	MoveClass	Duplicati.Library.Common.Platform.IsClientPosix
32	22	MoveClass	Duplicati.Library.Common.IO.Util.AppendDirSeparator

Local search based on problematic couplings

- Illustrative of what we're working towards
- Not yet what we'd consider a "good" solution, but encouraging

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

Current Work – Multi-objective Search

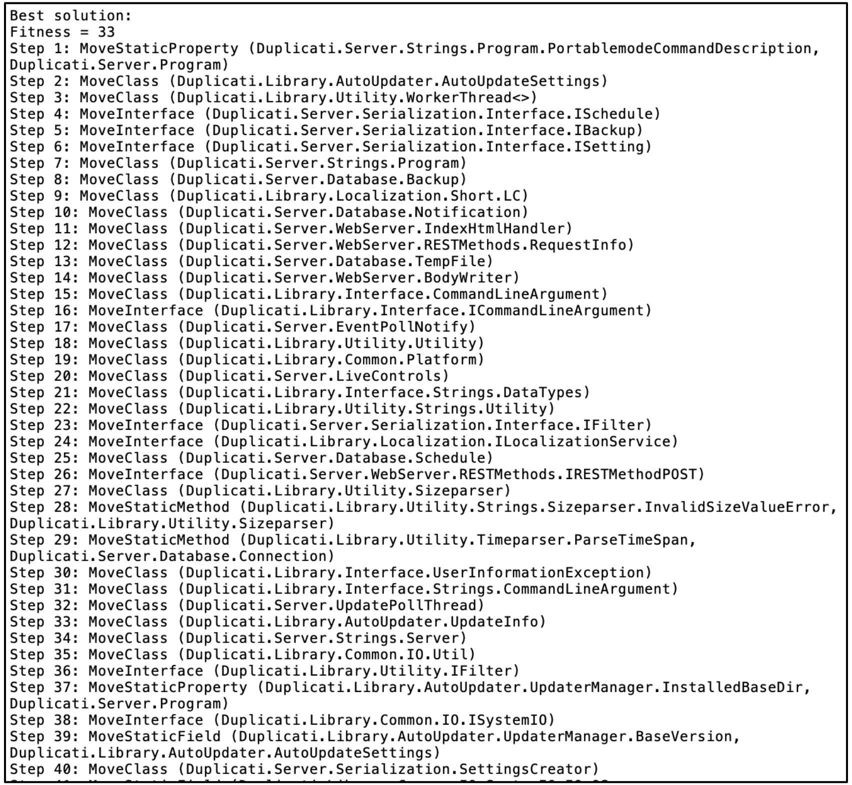
Multi-objective genetic algorithms like NSGA-II allow us to employ multiple fitness functions and generate Pareto-optimal solutions.

We are exploring fitness functions to find a combination that yields *recommendations that developers will accept*.

Candidate fitness functions include

- solving the core problem – minimizing problematic couplings
- reducing work – minimizing code changes and unrealized interfaces
- maintainable code – improving a range of code quality metrics
- understandable code – maximizing semantic coherence

Refactoring Recommendations



```
Best solution:
Fitness = 33
Step 1: MoveStaticProperty (Duplicati.Server.Strings.Program.PortablemodeCommandDescription,
Duplicati.Server.Program)
Step 2: MoveClass (Duplicati.Library.AutoUpdater.AutoUpdateSettings)
Step 3: MoveClass (Duplicati.Library.Utility.WorkerThread<>)
Step 4: MoveInterface (Duplicati.Server.Serialization.Interface.ISchedule)
Step 5: MoveInterface (Duplicati.Server.Serialization.Interface.IBackup)
Step 6: MoveInterface (Duplicati.Server.Serialization.Interface.ISetting)
Step 7: MoveClass (Duplicati.Server.Strings.Program)
Step 8: MoveClass (Duplicati.Server.Database.Backup)
Step 9: MoveClass (Duplicati.Library.Localization.Short.LC)
Step 10: MoveClass (Duplicati.Server.Database.Notification)
Step 11: MoveClass (Duplicati.Server.WebServer.IndexHtmlHandler)
Step 12: MoveClass (Duplicati.Server.WebServer.RESTMethods.RequestInfo)
Step 13: MoveClass (Duplicati.Server.Database.TempFile)
Step 14: MoveClass (Duplicati.Server.WebServer.BodyWriter)
Step 15: MoveClass (Duplicati.Library.Interface.CommandLineArgument)
Step 16: MoveInterface (Duplicati.Library.Interface.ICommandLineArgument)
Step 17: MoveClass (Duplicati.Server.EventPollNotify)
Step 18: MoveClass (Duplicati.Library.Utility.Utility)
Step 19: MoveClass (Duplicati.Library.Common.Platform)
Step 20: MoveClass (Duplicati.Server.LiveControls)
Step 21: MoveClass (Duplicati.Library.Interface.Strings.DataTypes)
Step 22: MoveClass (Duplicati.Library.Utility.Strings.Utility)
Step 23: MoveInterface (Duplicati.Server.Serialization.Interface.IFilter)
Step 24: MoveInterface (Duplicati.Library.Localization.ILocalizationService)
Step 25: MoveClass (Duplicati.Server.Database.Schedule)
Step 26: MoveInterface (Duplicati.Server.WebServer.RESTMethods.IRESTMethodPOST)
Step 27: MoveClass (Duplicati.Library.Utility.Sizeparser)
Step 28: MoveStaticMethod (Duplicati.Library.Utility.Strings.Sizeparser.InvalidSizeValueError,
Duplicati.Library.Utility.Sizeparser)
Step 29: MoveStaticMethod (Duplicati.Library.Utility.Timeparser.ParseTimeSpan,
Duplicati.Server.Database.Connection)
Step 30: MoveClass (Duplicati.Library.Interface.UserInformationException)
Step 31: MoveClass (Duplicati.Library.Interface.Strings.CommandLineArgument)
Step 32: MoveClass (Duplicati.Server.UpdatePollThread)
Step 33: MoveClass (Duplicati.Library.AutoUpdater.UpdateInfo)
Step 34: MoveClass (Duplicati.Server.Strings.Server)
Step 35: MoveClass (Duplicati.Library.Common.IO.Util)
Step 36: MoveInterface (Duplicati.Library.Utility.IFilter)
Step 37: MoveStaticProperty (Duplicati.Library.AutoUpdater.UpdaterManager.InstalledBaseDir,
Duplicati.Server.Program)
Step 38: MoveInterface (Duplicati.Library.Common.IO.ISystemIO)
Step 39: MoveStaticField (Duplicati.Library.AutoUpdater.UpdaterManager.BaseVersion,
Duplicati.Library.AutoUpdater.AutoUpdateSettings)
Step 40: MoveClass (Duplicati.Server.Serialization.SettingsCreator)
```

Our prototype generates recommendations as a sequence of refactorings

- clear directions for a developer
- independently reviewable prior to changing code
- built on refactorings supported by development environments
- future potential to automate application of the refactorings to code

Wrap Up

Please let us know if you are interested. For example, if you

- would like to discuss potential use for estimation
- have ideas on important objectives to consider
- have access to data we could use to validate the research
- would like to discuss opportunities to pilot the refactoring recommendations

Our current prototype

- is open source, but relies on one commercial tool
- handles C# code, with future potential for Java
- has been tested on up to 1.2M SLOC code bases

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.

For More Information

James Ivers

Initiative Lead
Architecture Analysis, Design, and Automation
jivers@sei.cmu.edu
412-268-7793

© 2019 Carnegie Mellon University

[Distribution Statement A] Approved for public release and unlimited distribution.