AI for Software Engineering

Ipek Ozkaya and James Ivers

August 3, 2021

SEI Educator's Workshop

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM21-0688

AI for Software Engineering (AI4SE): A Blessing or a Curse?

AI4SE has become an umbrella term to refer to research that uses AI approaches to tackle software engineering challenges.

- AI approaches can improve developer tools to eliminate subtle mistakes that later become hard to detect and propagate fixes for.
  - e.g. Github Copilot by Microsoft, “AI pair programmer”

  Pros: saving developer time, improved correctness over time

  Cons: incorrect examples, licensing implications, violate copyrights
- Creating appropriate data sets has also emerged as one of the research areas in AI4SE
  - e.g. Project Codenet by IBM (https://arxiv.org/abs/2105.12655)

What will application of AI help solve that other approaches to date have not been able to help improved automated support for developers?

Can AI Help Solve Enduring Challenges of Software Engineering: Better, Faster, Cheaper?

Streamlining software development tasks towards successful system delivery continues to be resource intensive and error prone.

We expect developers to grasp and manage ripple effects in increasingly complex (due to size, distribution, incompatibility, …) systems without effective tool support.

Lacking effective automation, time spent in design and testing continue to be reduced first when schedule challenges hit, further jeopardizing the resulting quality of the systems deployed.

System sustainment and evolution, especially for legacy systems, continue to be a labor intensive, and high-risk effort.

Conformance to quality standards and intended architectures are not guaranteed as part of the software development frameworks and tool chains.

Common theme: Are we providing effective tools to improve developers tasks and cognitive overload towards developing higher quality software?

Shaping Research in AI4SE

Focus on small, frequent, and AI-relevant problems

- Do what developers already do more efficiently (e.g., test faster).
  - N. Alshahwan, X. Gao, M. Harman, Y. Jia, K. Mao, A. Mols, T. Tei, I. Zorin: *Deploying Search Based Software Engineering with Sapienz at Facebook.* SSBSE 2018: 3-45
- Do what developers already do better (e.g., catch more bugs).
  - NC Shrikanth, T Menzies: *The Early Bird Catches the Worm: Better Early Life Cycle Defect Predictors*. https://arxiv.org/abs/2105.11082 2021.
- Integrate things that are currently disconnected (e.g. requirement traceability).
  - Jinfeng Lin, Yalin Liu, Qingkai Zeng, Meng Jiang, Jane Cleland-Huang: Traceability Transformed: Generating more Accurate Links with Pre-Trained BERT Models. ICSE 2021: 324-335
- Teach developers how to do tasks better as they go (e.g., advise/mentor with real-time feedback on implementation errors).
  - Anshul Gupta, Neel Sundaresan, Intelligent code reviews using deep learning KDD'18 Deep Learning Day, August 2018, London, UK
- Do tasks developers aren't able to do today (e.g., leverage new data to integrate new conformance checks or generate new tests).
  - Ongoing SEI work
- Scale and optimize what developers already can do (e.g., consider more alternative design options).
  - Ongoing SEI work

Polling Question 1

Shaping Research in AI4SE – Research Challenges

Are developers better at writing specifications (AI generates code) than writing code (AI improves code)?

What new software development data needs to be collected (ethically also ensuring security and privacy) to enable future AI4SE research?

How can developer trust be established?

What does a human-computer AI4SE “partnership” look like?

- Intern who I don't entirely trust, but who does save me a lot of time?
- Bot that does things for me?
- Partner that advises me?

What new and augmented activities become part of the software development lifecycle (SDLC), in an AI-assisted paradigm?

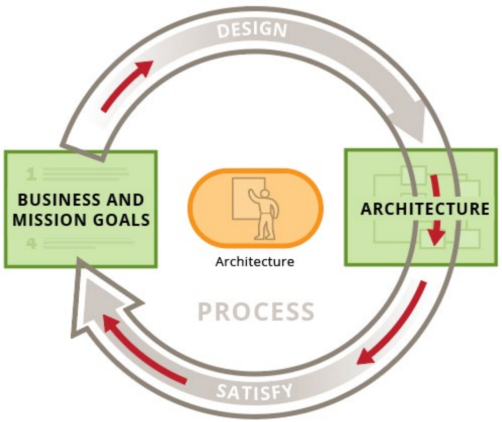
Applying AI4SE Approach to Software Architecture Challenges

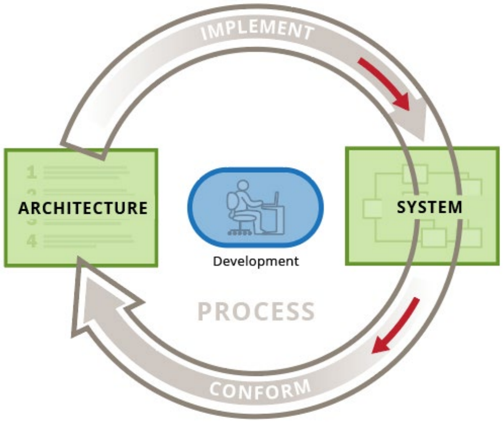
Software architecture is an important abstraction that helps organizations satisfy a wide range of business and mission goals.

- A significant portion of SEI stakeholders deal with large-scale changes to existing systems (e.g., modernization)
- A common impediment is that architecture and design documentation is often missing or out of date

When architecture and design information differ from code, we generally

- Trust the code
- Lose the ability to apply architectural analyses (e.g., diagnosing root causes or the implications of a potential change)





How Can AI for Software Engineering Help?

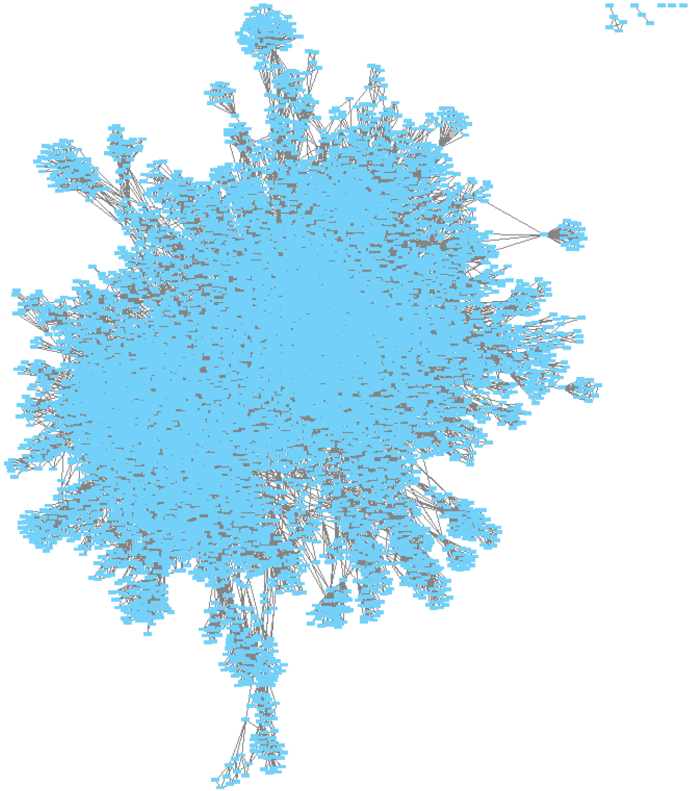
We are motivated to help create a new generation of automation for architects that helps bridge the gap between architecture abstractions and code.

Two SEI projects are currently investigating applications of AI to

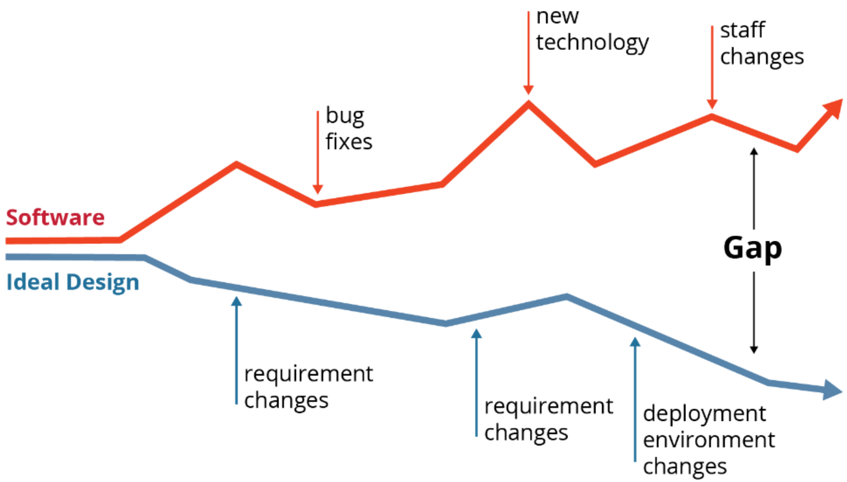
- Refactor code to improve its design
- Check that implementations conform to "as intended" designs

Current SEI Research

Untangling the Knot



Software Is Never Done



Change is inevitable

- Requirements change
- Business priorities change
- Programming languages change
- Deployment environments change
- Technologies and platforms change
- Interacting systems change
- ...

To adapt to such changes, we need to periodically improve software structure (architecture) to match today's needs.

A Key Barrier to Software Evolution

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

- Reusing capability in a different system, 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

Automation that generates solutions can significantly reduce the cost and schedule impact of many kinds of software evolution.

An 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:

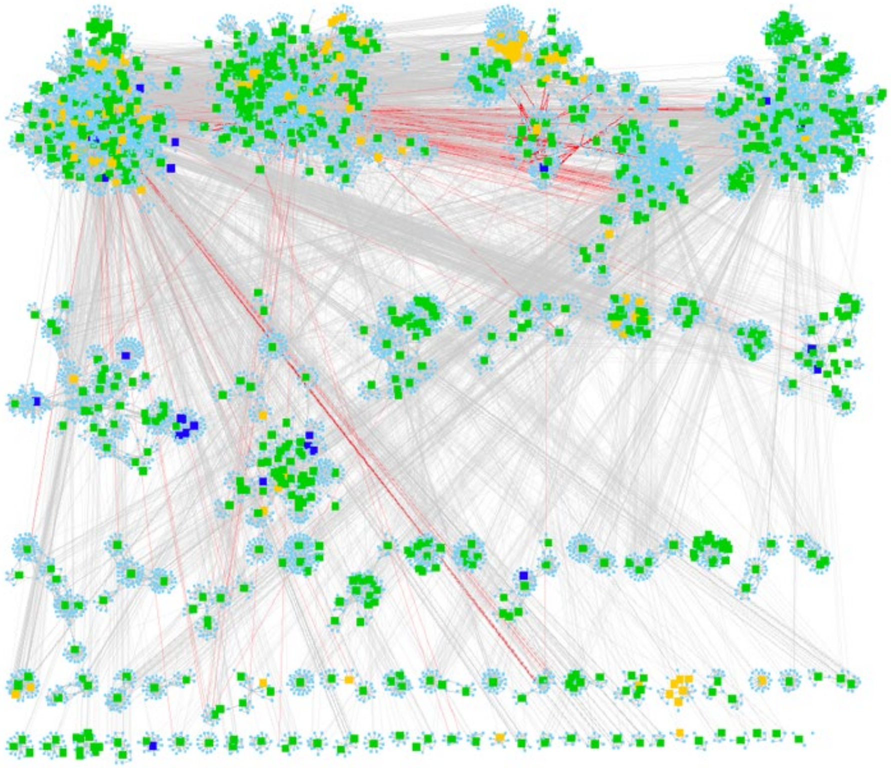
- Solves project-specific problems
- Uses a semi-automated approach
- Addresses all three labor-intensive activities
- Allows refactoring to be completed in less than 1/3 of the time required by manual approaches

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

J. Ivers, I. Ozkaya, R. L. Nord, C. Seifried. **Next Generation Automated Software Evolution: Refactoring at Scale**. 2020. *28th Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20).*

Key Concept – Problematic Couplings



Only certain software dependencies interfere with any particular goal.

For example, if we want to harvest a feature:

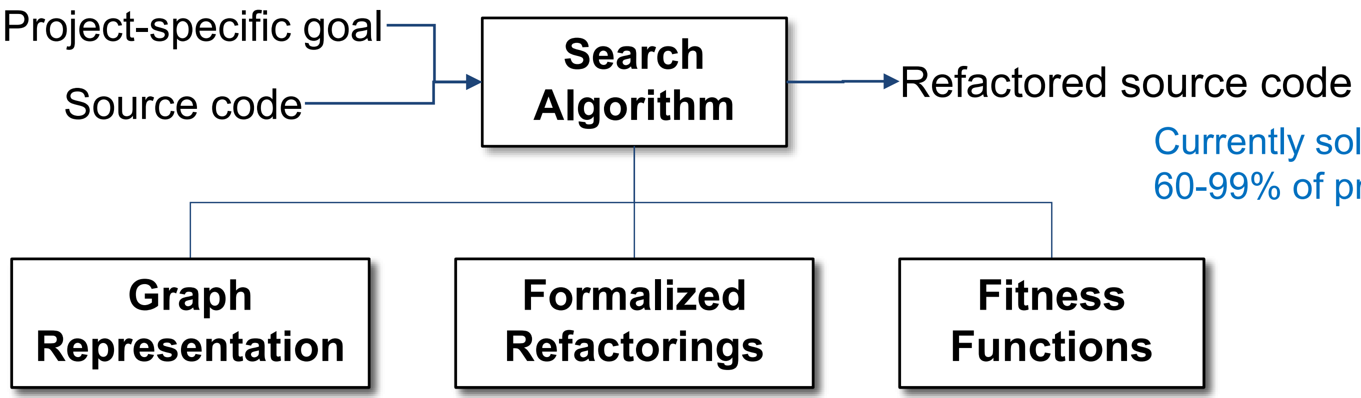
- The core problem is dependencies (red lines) from software being harvested to software that is being left behind
- All other dependencies are irrelevant to the goal, allowing us to focus our analysis and search for solutions

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

Our Approach

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

Tested to 1.2M SLOC of C#



Currently solving 60-99% of problem

Uses static code analysis to generate an intermediate representation

Uses the graph for pre-conditions and transformations for Fowler-style refactorings

Measures computed on the graph to judge "goodness" of solutions

Satisfying Multiple Criteria

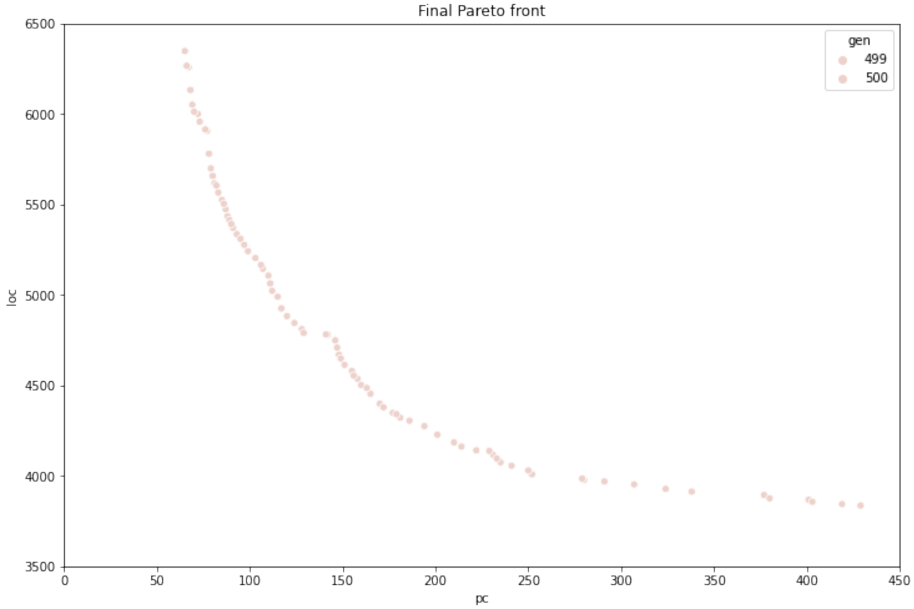
We use a combination of fitness functions to generate *recommendations that developers will accept*.

Our prototype uses a **multi-objective genetic algorithm**, based on NSGA-II, to generate Pareto optimal solutions that represent different trade-offs among objectives.

Examples include

- solution to the core problem – minimizing problematic couplings
- less work – minimizing code changes and unrealized interfaces
- maintainable code – improving code quality metrics
- understandable code – maximizing semantic coherence
- secure code – minimizing public members

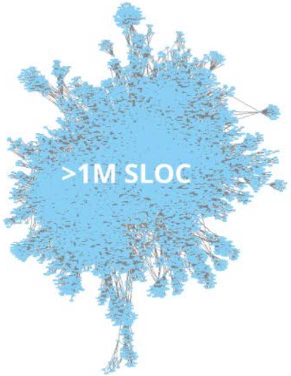
Pareto-optimal Solutions

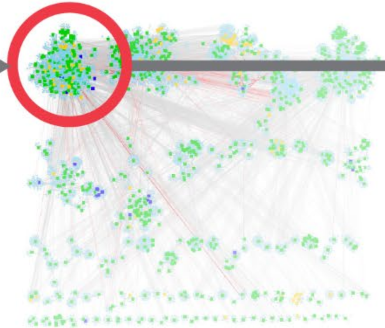


Multi-objective optimization generates **choices** that represent trade-offs among competing objectives.

- This search used two objectives – problematic couplings and lines of code.
- Search is able to make significant progress, reducing problematic couplings to 23% of the original measure.
- It's a reasonable Pareto front; options indicate distinct trade-offs.
- It includes a number of solutions that are likely to be considered impractical, though this is subjective.

Generating Refactoring Recommendations





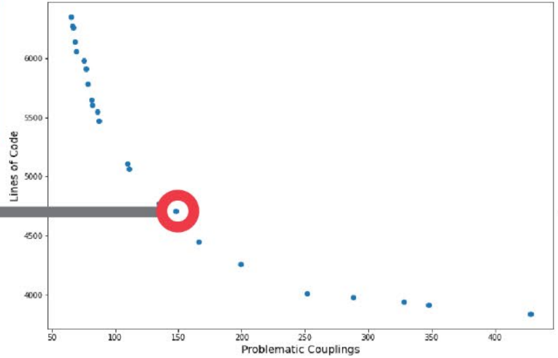
Select Objectives

- 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)

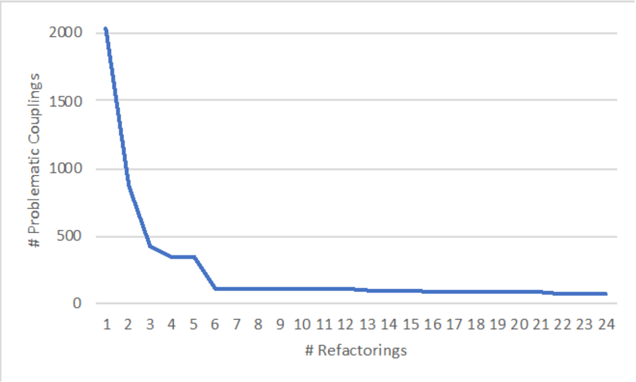
```
Best solution:
Fitness = 33
Step 1: MoveStaticProperty (Duplicati.Server.Strings.Program.PortablemodeCommandDe-
scription, Duplicati.Server.Program)
Step 2: MoveClass (Duplicati.Library.AutoUpdater.AutoUpdateSettings)
Step 3: MoveClass (Duplicati.Library.Utility.WorkerThread<>)
Step 4: MoveInterface (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.InvalidSizeVal-
ueError, 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)
```

Select and implement a solution that suits your context.



Refactoring Recommendations

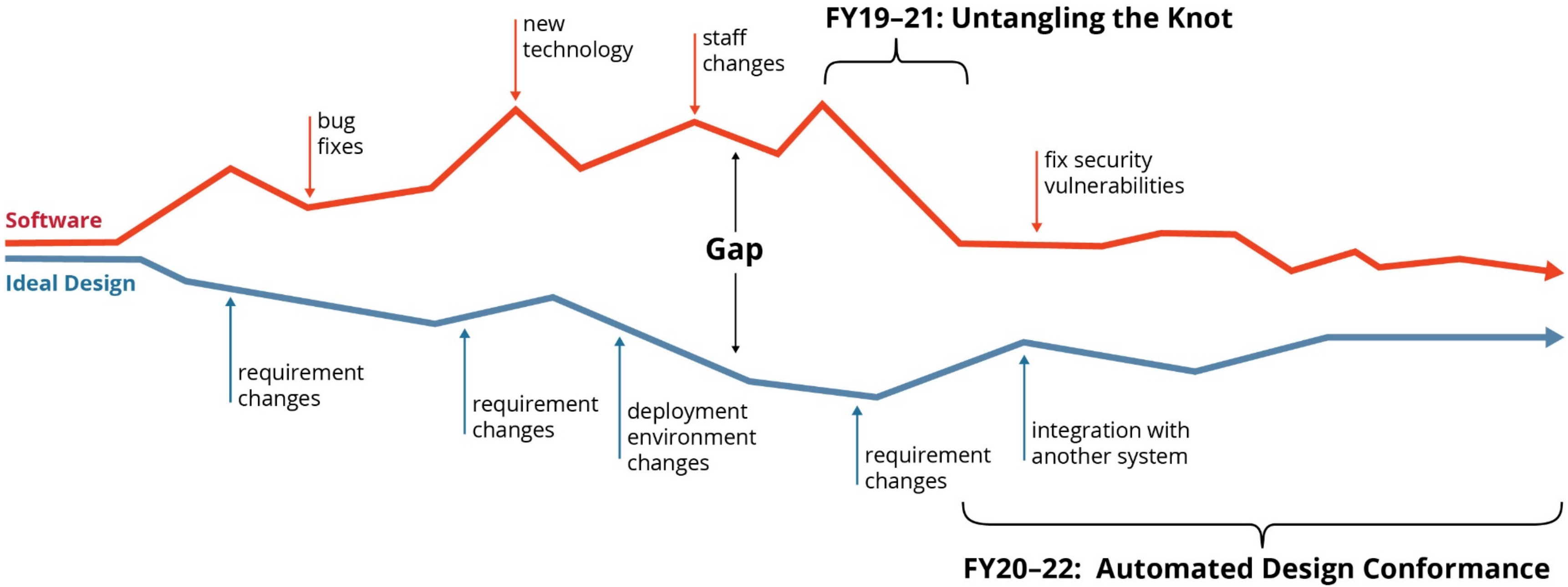
```
Best solution:
Fitness = 33
Step 1: MoveStaticProperty (Duplicati.Server.Strings.Program.PortablemodeCommandDescription,
Duplicati.Server.Program)
Step 2: MoveClass (Duplicati.Library.AutoUpdater.AutoUpdateSettings)
Step 3: MoveClass (Duplicati.Library.Utility.WorkerThread<>)
Step 4: MoveInterface (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 to code

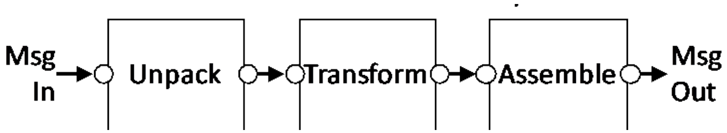
Vision: AI for Software Engineering Can Help Keep Software Aligned with Needs



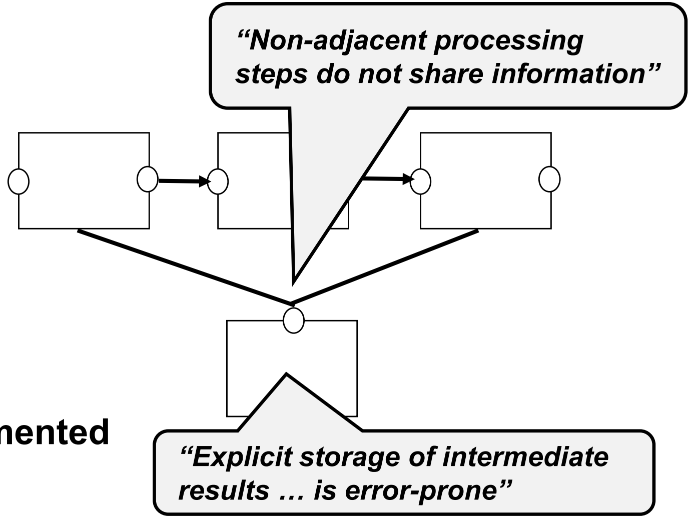
J. Ivers, I. Ozkaya, R. L. Nord. **Can AI Close the Design-Code Abstraction Gap?** Software Engineering Intelligence Workshop 2019, co-located with Intl. Conference on Automated Software Engineering: 122-125.

Current SEI Research

Automated Design Conformance

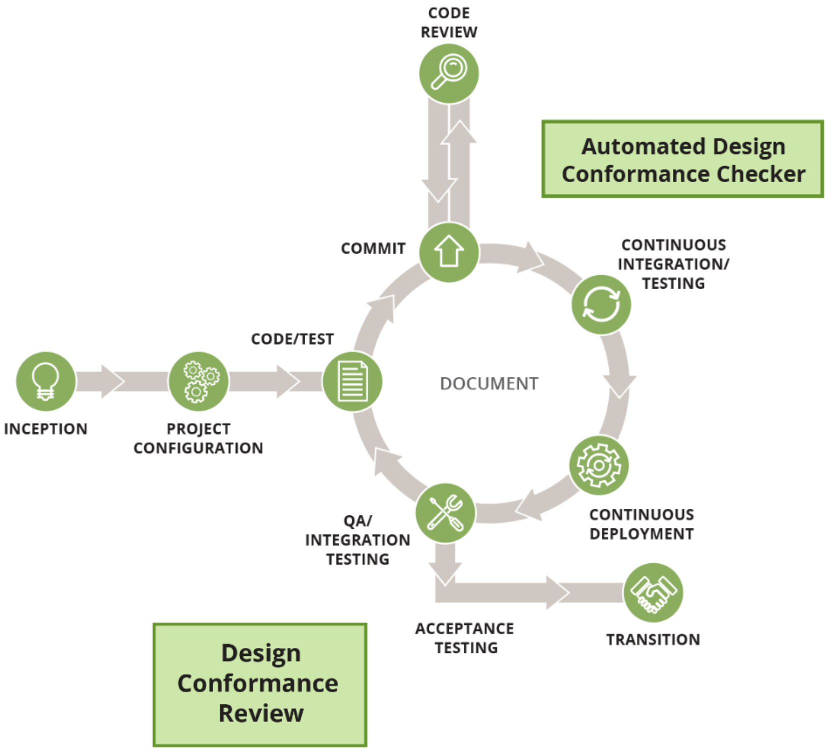


intended



implemented

Automated Design Conformance during CI

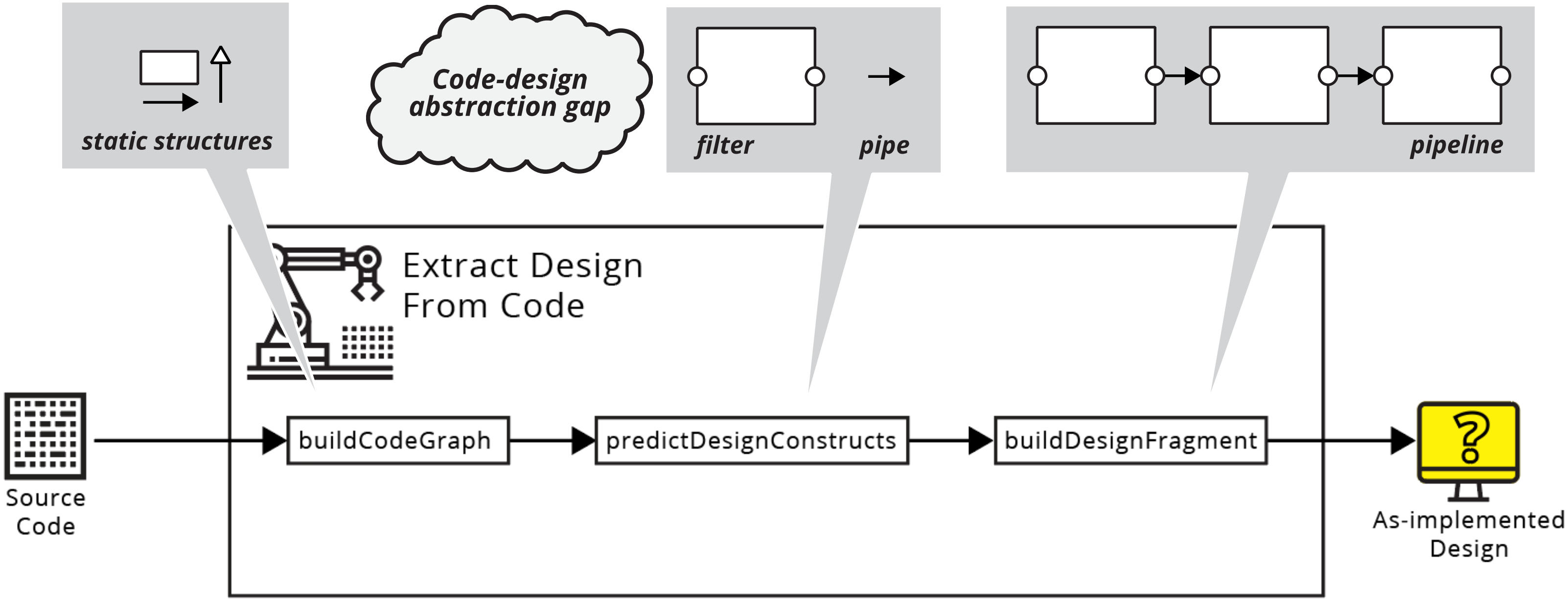


An automated design conformance checker integrated into a continuous integration workflow will reduce time to detect violations from months or years to hours.

Automation enables early detection and allows remediation before the violation gets “baked in” to the implementation.

Detection of nonconformances allows program managers to hold developers (contractor or organic) accountable.

Code-Design Abstraction Gap



Ivers et al. (2019). **Can AI Close the Design-Code Abstraction Gap?** *International Workshop on Software Engineering Intelligence, IEEE/ACM International Conference on Automated Software Engineering (ASE).*

From Code to Design Fragment

How do you recognize design abstractions from code?

- Rules or classifiers?
- Based on what data?
- How generalizable can you get?

Hotspot (Qt)
github.com/KDAB/hotspot

- **8K** code lines
- **2,648** nodes and **11,427** relations
- **7** publishers, **37** subscribers