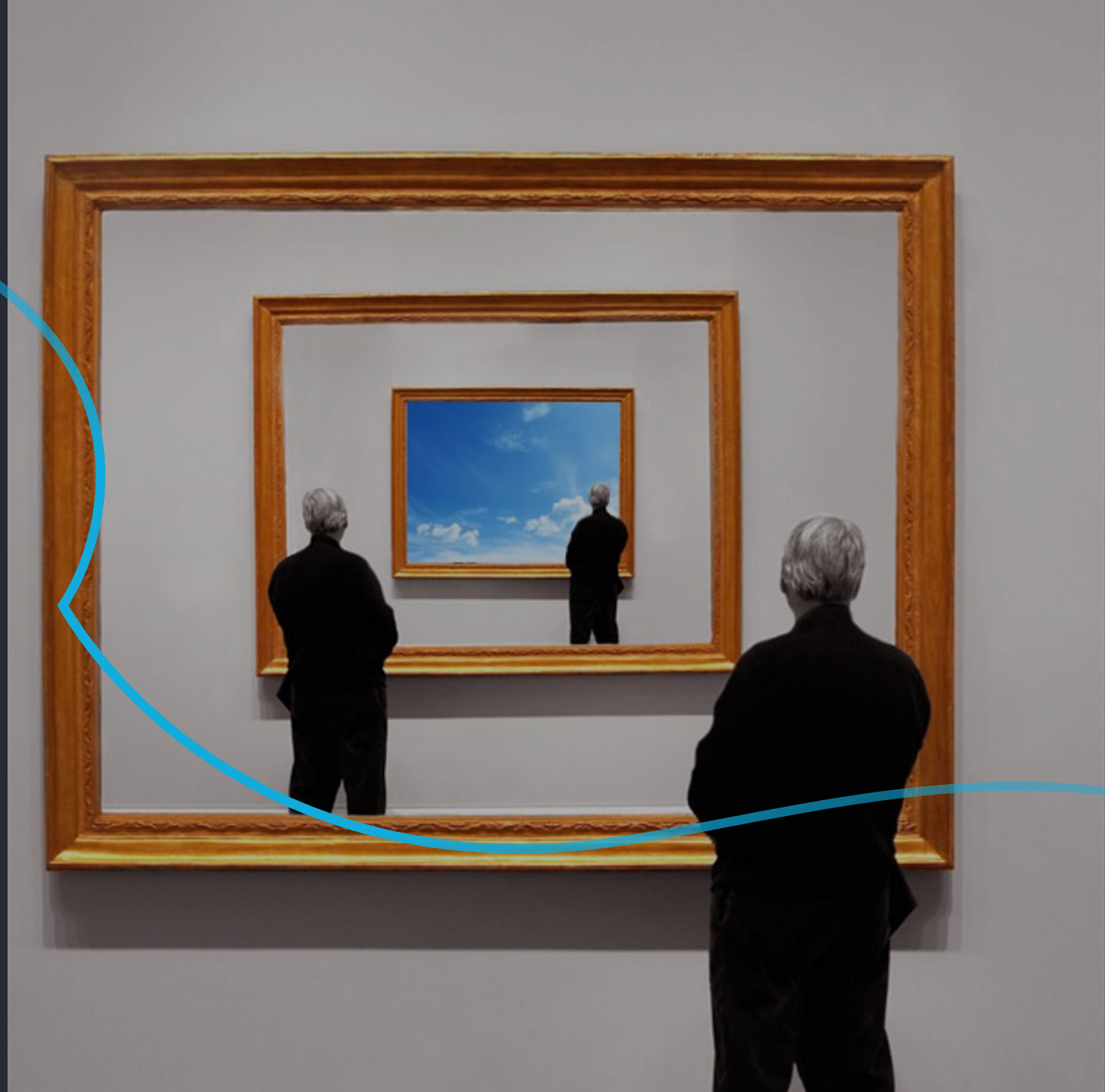
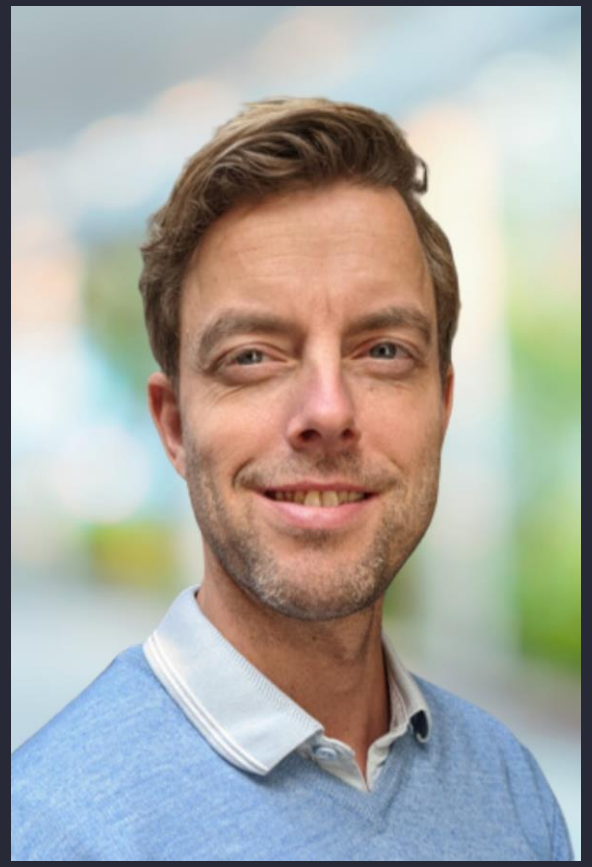
Capgemini engineering

HOW TO CONTINUOUSLY AND EFFICIENTLY REJUVENATE A SOFTWARE CODE BASE?

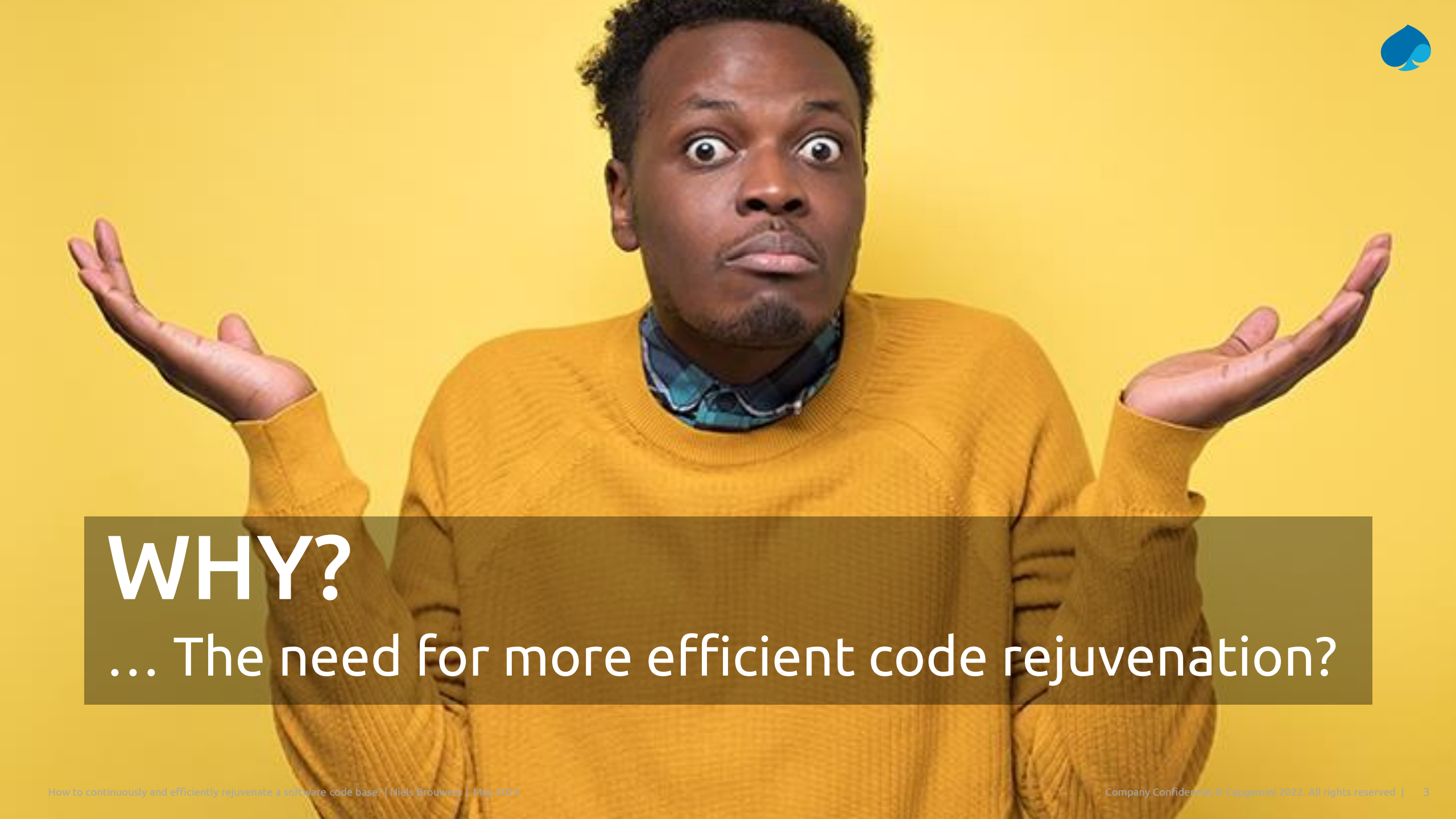


May 16th, 2024 | 040coders | Niels Brouwers

NIELS BROUWERS – SOLUTION ARCHITECT

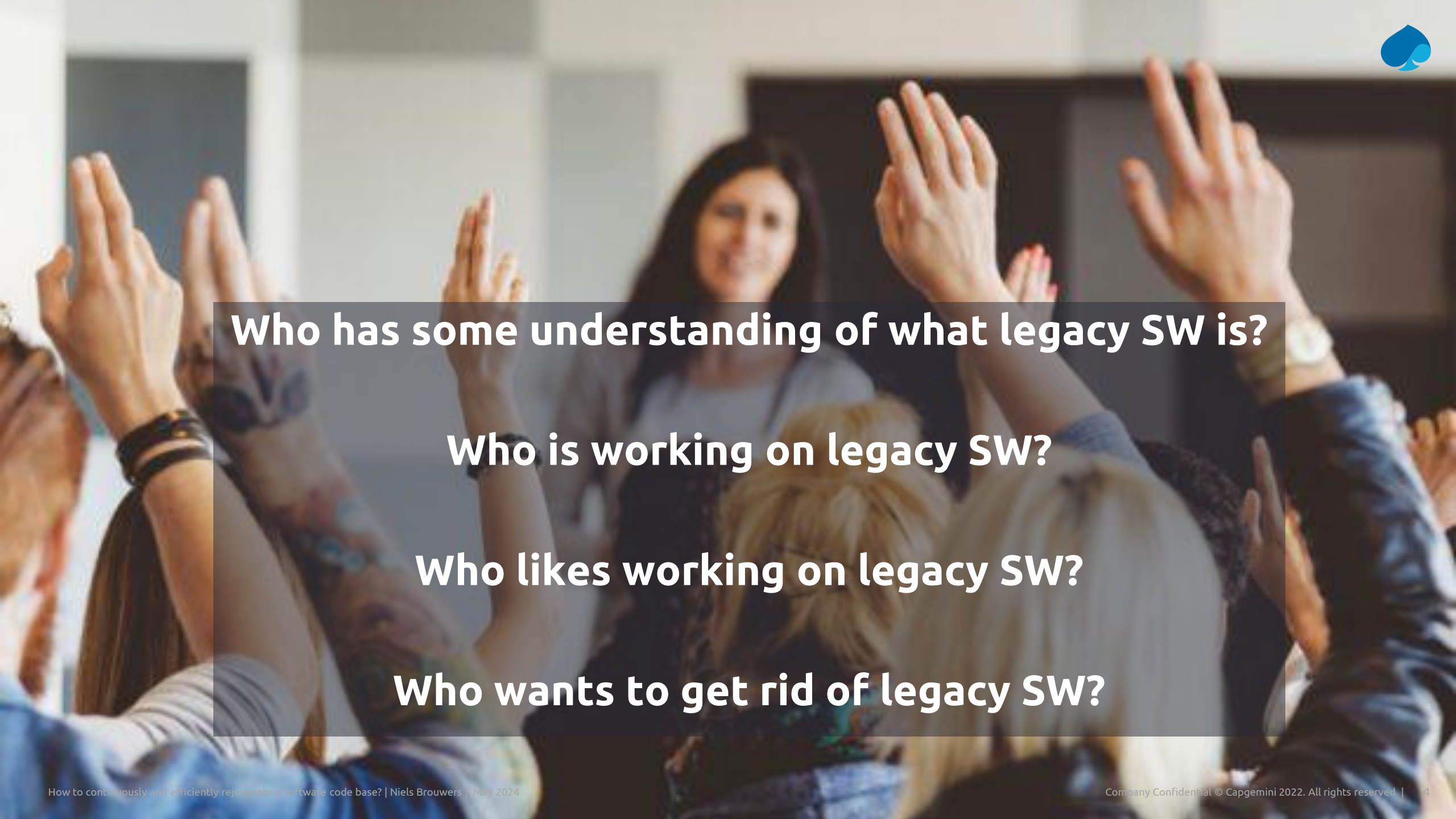


- Accelerate software development
- Increasing the level of abstraction
- Intensifying the level of automation
- Model-driven engineering (MDE)
- Automated software rejuvenation



WHY?

… The need for more efficient code rejuvenation?



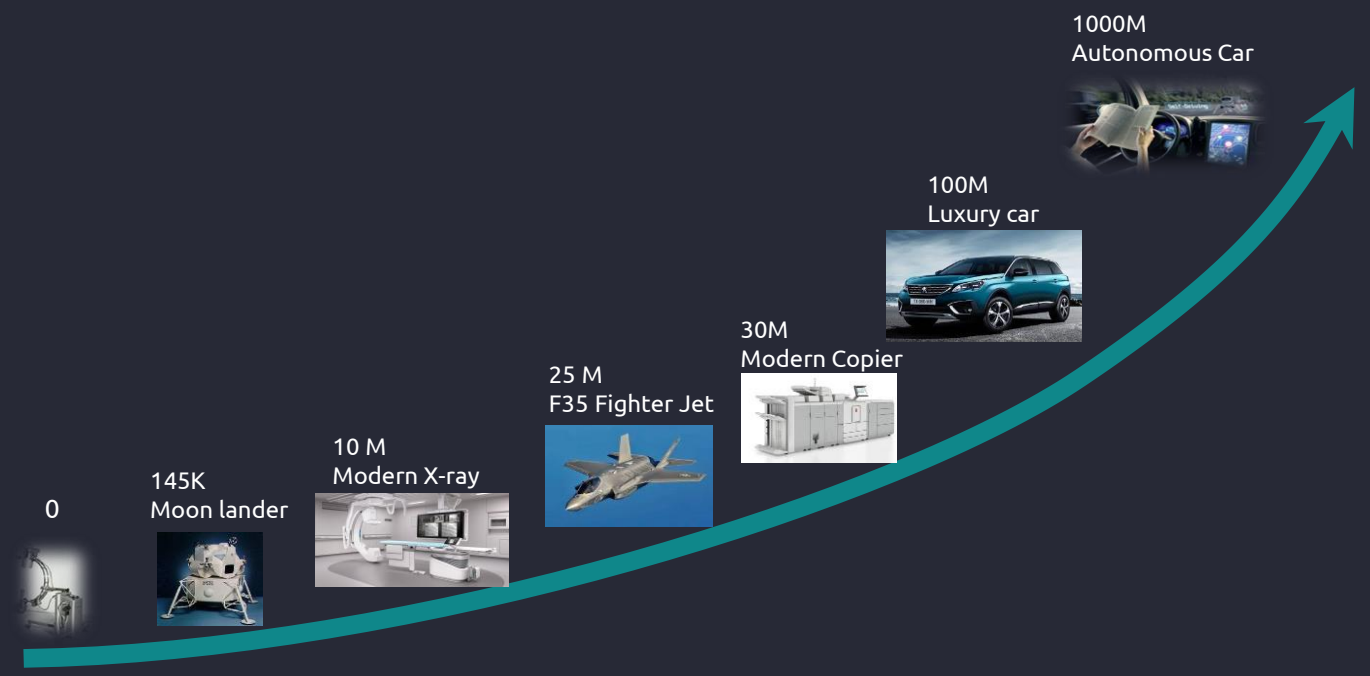
Who has some understanding of what legacy SW is?

Who is working on legacy SW?

Who likes working on legacy SW?

Who wants to get rid of legacy SW?

LEGACY SW?

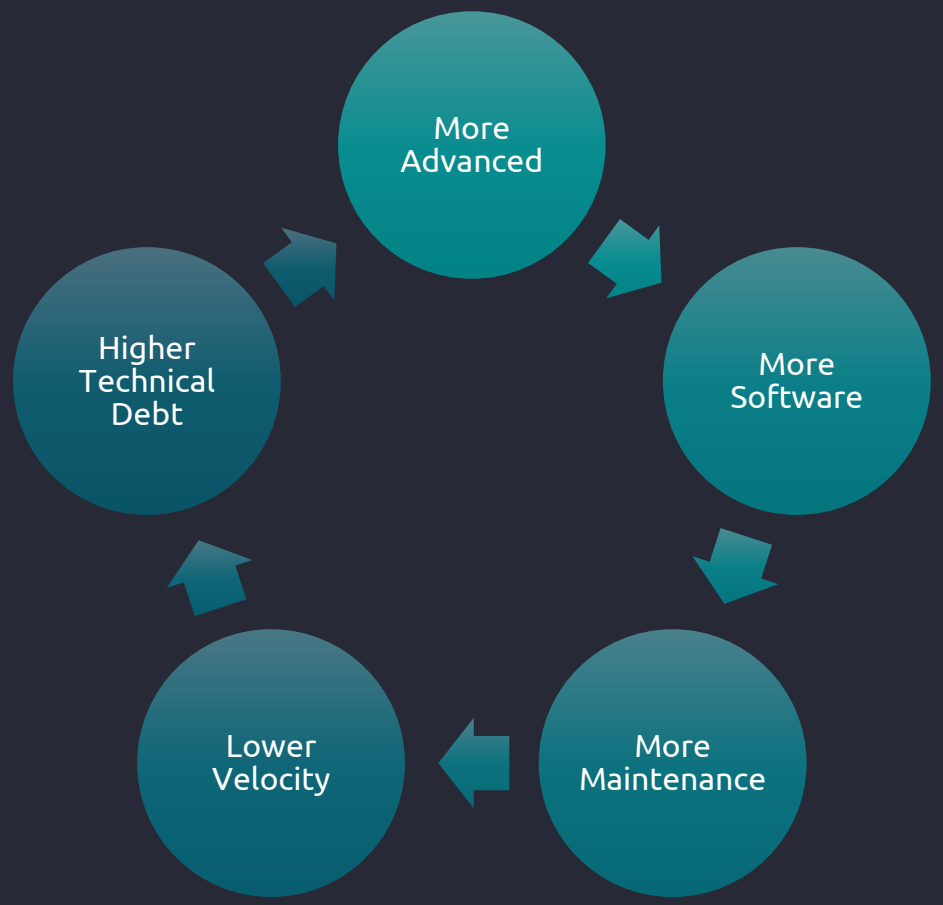


- ☑ Deteriorated software quality
- ☑ Outdated technologies
- ☑ Dead and obsolete code
- ☑ Lack of knowledge

"Software that notably resists modification or evolution"

HOW LEGACY SOFTWARE (INEVITABLY?) IS CREATED OVER TIME

Accumulation of technical debt, to the point where software resists being modified

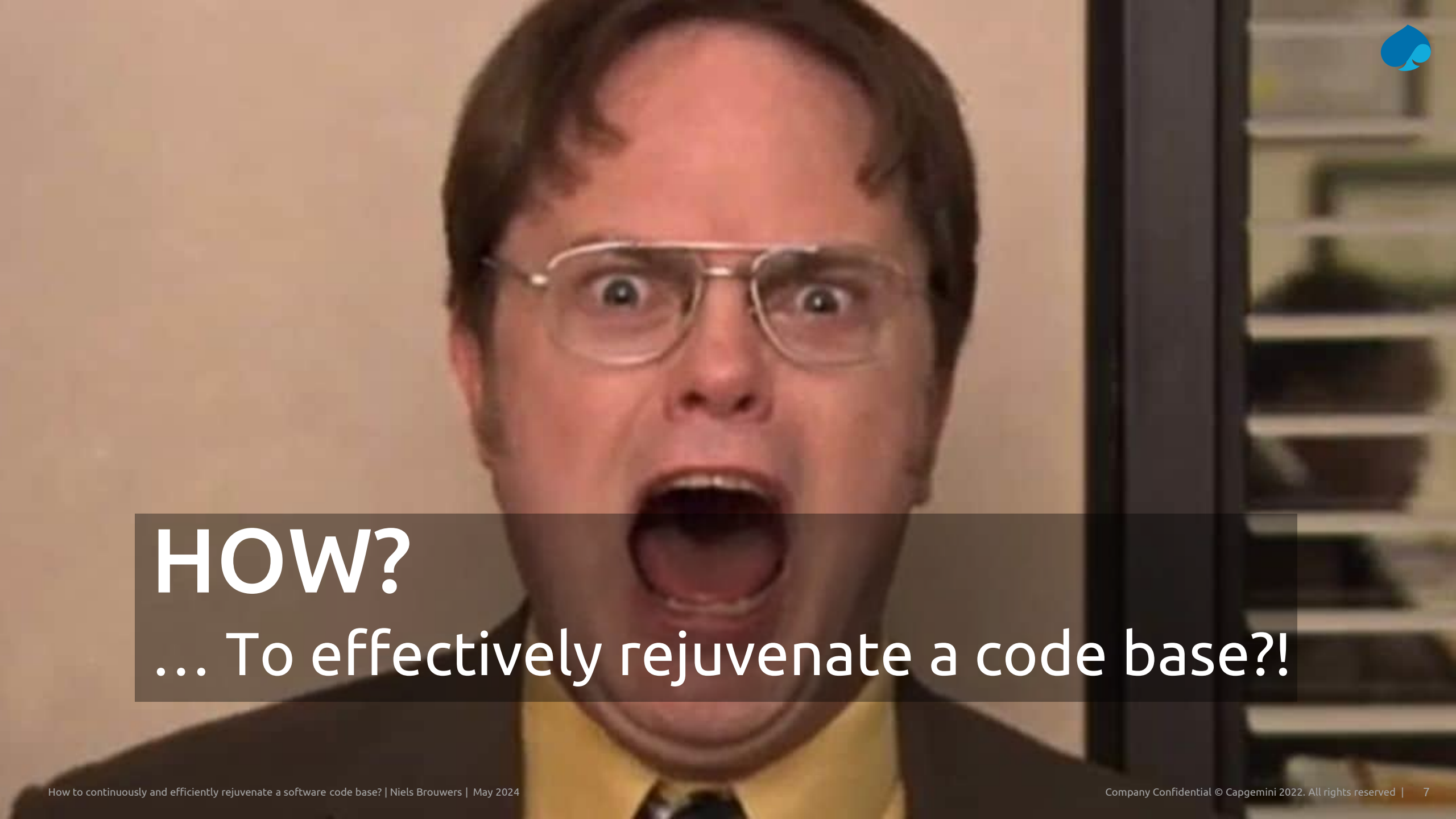


Each full-time developer can maintain “only” 50KLoc[1]

Many companies don’t have “luxury” to spend 200 FTE to maintain a 10 MLoC code base

Prevent legacy SW by performing maintenance more efficiently!

[1] Wayne Lobb e.a “Software Development and Maintenance Effort/Cost Models”, Foilage whitepaper

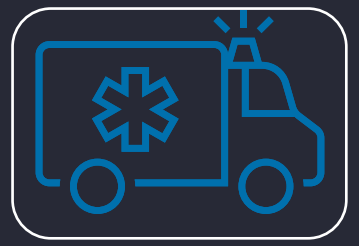


HOW?

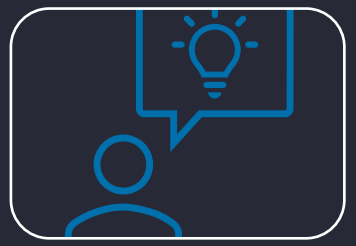
… To effectively rejuvenate a code base?!

SOLUTION: AUTOMATE SOFTWARE MAINTENANCE …

Software Maintenance Tasks

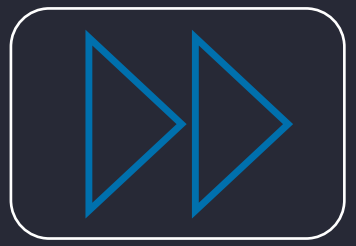


Solve architecture or code issues



Improve code readability

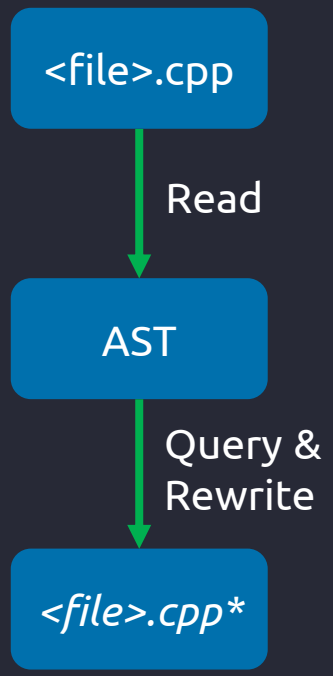
Remove obsolete and dead code



Migrate to improved API

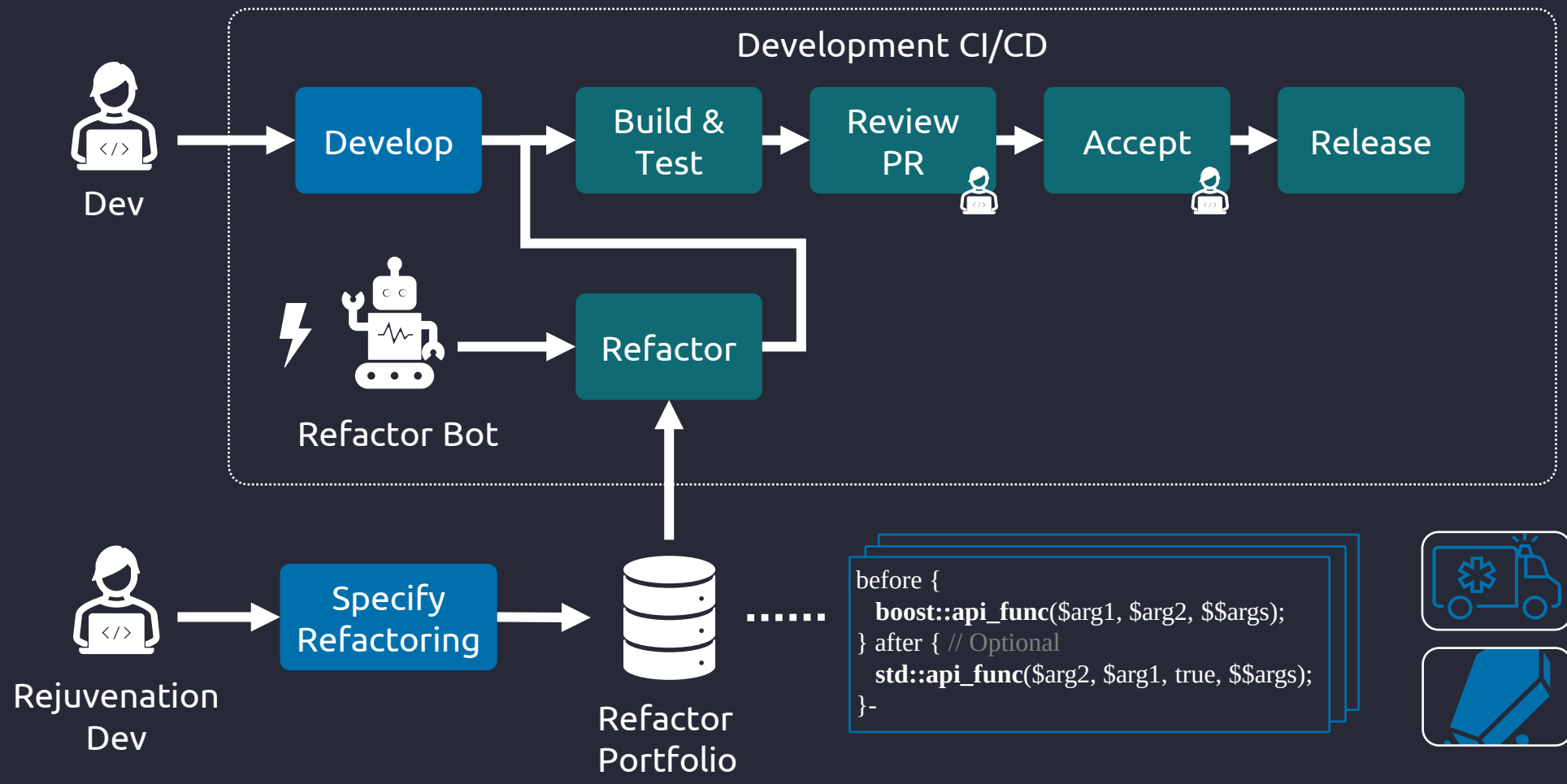
generalize

Replacement of code patterns, automatable!

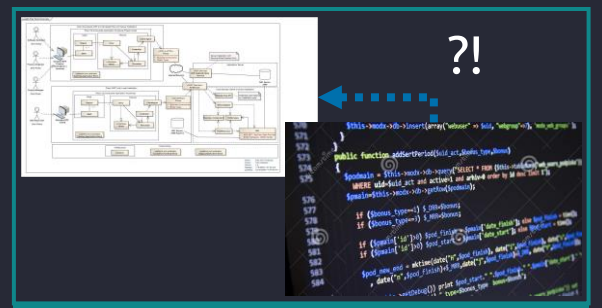



```
before {
  boost::foo($arg1, $arg2, $$args);
} after { // Optional
  std::foo($arg2, $arg1, true, $$args);
}
```

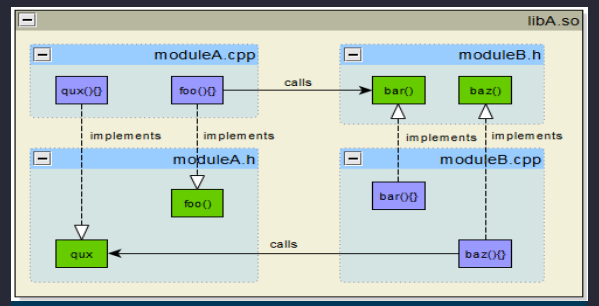

… CONTROLLED & RISK-FREE WHEN INTEGRATED INTO CI/CD …



… FOLLOWING AN AUTOMATED AND CONTINUOUS PROCESS …

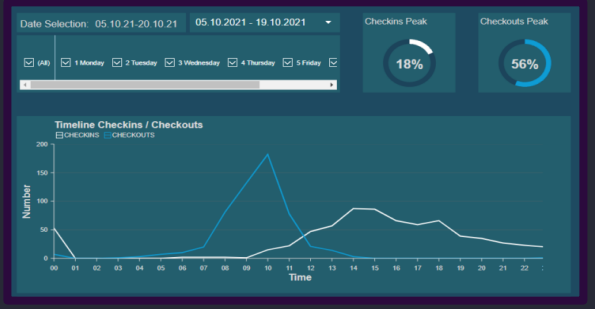


4. Prevent
“No more!”



1. Analyze
“Where?”

2. Monitor & Plan
“When?”



3. Refactor
“Do it”

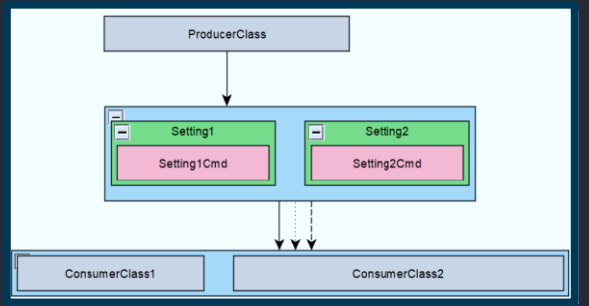
```
before {
  boost::api_func($arg1, $arg2, $$args);
} after { // Optional
  std::api_func($arg2, $arg1, true, $$args)
}-
```

Merge pull request

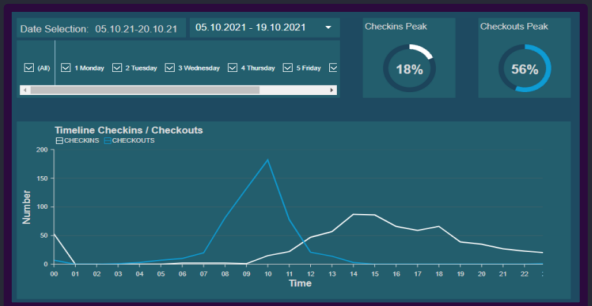
WE INTERRUPT THIS PROGRAM FOR A

COMMERCIAL BREAK

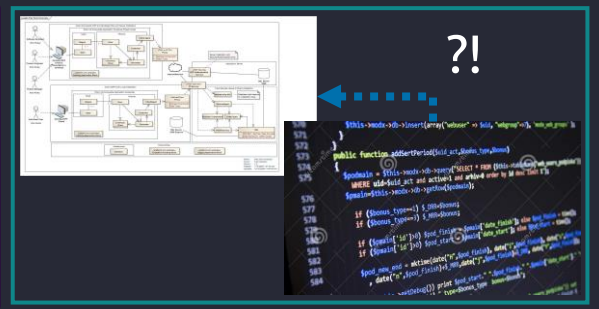
… ENABLED BY R.E.B.O.R.N.



Analysis
"Where?"



Reporting
"When?"



Quality Guarding
"Prevent!"

```
before {
  boost::api_func($arg1, $arg2, $$args);
} after { // Optional
  std::api_func($arg2, $arg1, true, $$args)
}-
```

Merge pull request

R-Actions
"Here it is!"

Software Intelligence

- Extractors (code, project, architecture)
- CI/CD Integrated
- Centralized in Graph Database

Powered by:

ESI, neo4j, docker, yEd, Grafana

Supporting:

Ada, Java

WHAT?

… is happening under the hood,
and show me some use cases, please!

CODE REFACTORING AT SCALE, RIGOUROUS AND DETERMINISTIC

Transformations to improve quality and migrate technologies

<file>.cpp

Read

AST

Query & Rewrite

*<file>.cpp**

1

```
replaceName("FunctionCallExpression",
"boost::api_func"," std::api_func");
```

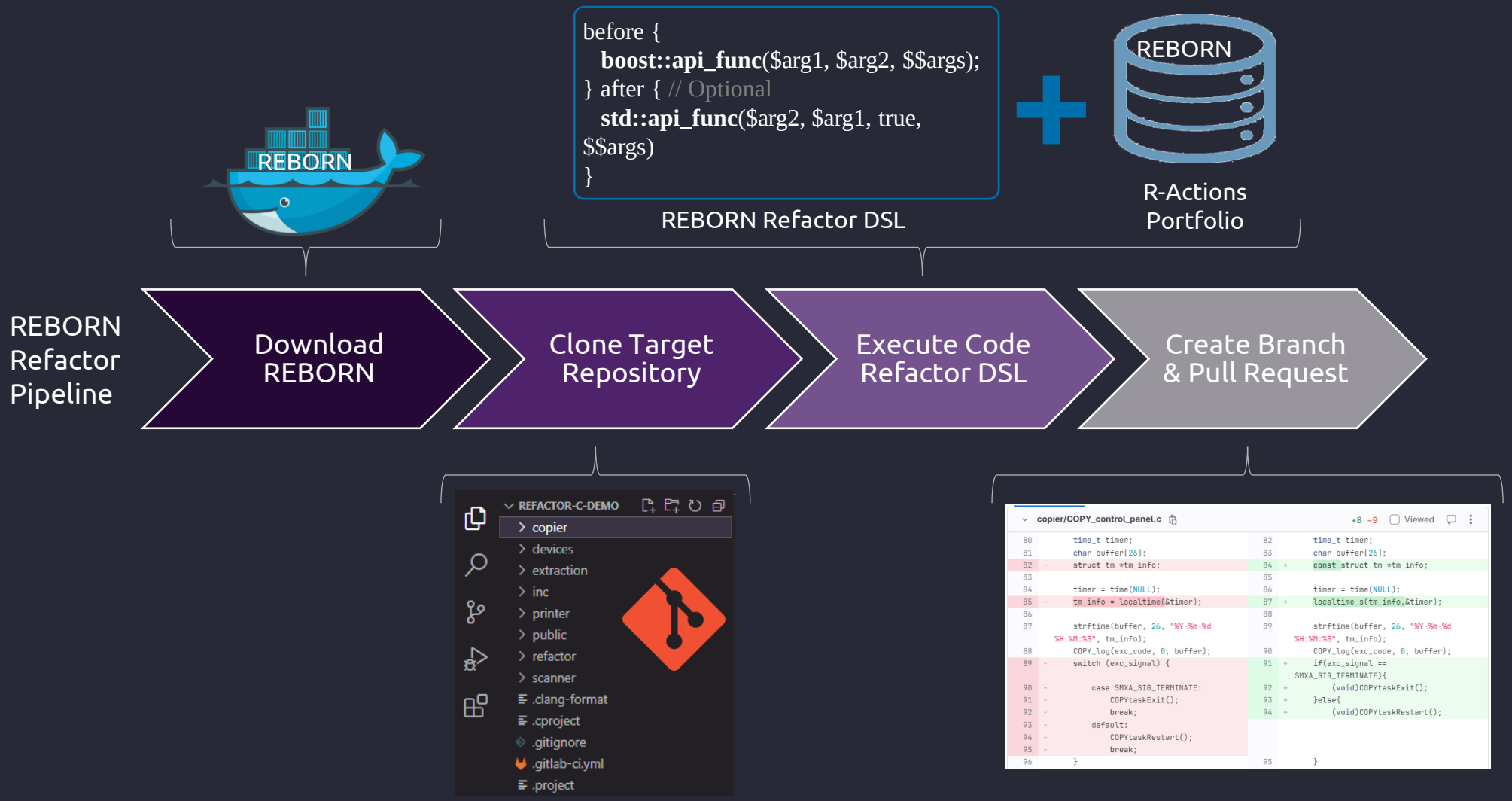
2

```
before {
   boost::api_func($arg1, $arg2, $$args);
} after { // Optional
   std::api_func($arg2, $arg1, true, $$args)
}
```

3

```
ASTFinder.findAll(atu, IASTFunctionCallExpression.class)
   .stream()
   .map(IASTFunctionCallExpression::getFunctionNameExpression)
   .filter(e->"boost::api_func".equals(e.toString()))
   .forEach(e->rewrite.replace(e, "std::api_func"));
```


DEMO: AUTOMATED CODE REFACTORING USING REBORN



SOME EXAMPLES OF AUTOMATED SOFTWARE MAINTENANCE

Solving code defects & smells

Convert test-suites to GoogleTest

Removal of dead and obsolete code

Make source- & project files self-contained

…

MIGRATION OF TEST SUITES FROM BOOST.TEST TO GOOGLE TEST

- Mainly standard refactoring using the refactor DSL were used.

```
replace{ LOTST_TEST_REQUIRE(!$value);}  with { ASSERT_FALSE($value); }
replace{ LOTST_TEST_REQUIRE($value);}   with { ASSERT_TRUE($value); }
replace{ BOOST_REQUIRE_MESSAGE(0 < $act, $value2);}     with { ASSERT_LT(0u, $act) << $value2; }
replace{ BOOST_REQUIRE_MESSAGE($exp < $act, $value2);}  with { ASSERT_LT($exp, $act) << $value2; }
replace{ BOOST_REQUIRE_MESSAGE($exp == $act, $value2);} with { ASSERT_EQ($exp, $act) << $value2; }
replace{ LOTST_CHECK_MESSAGE($exp == $act, $value2);}   with { ASSERT_EQ($exp, $act) << $value2; }
replace{ LOTST_CHECK_MESSAGE($exp != $act, $value2);}   with { ASSERT_NE($exp, $act) << $value2; }
```

- Complex cases were handled by a specific method that directly manipulates the AST. For instance:
  - migrateTestClassToGoogleTest();
  - removeTestMainFromProject();

MIGRATION OF TEST SUITES FROM PROPRIETARY TO GOOGLE TEST

Proprietary Testing

```
#include statements
#include "ExampleTestCase.h"
#include <Windows.h>

static constant example;

Test* ExampleTestCase::suite()
{
    TestSuite* testSuite = new TestSuite();

    testSuite->add(new TestCaller<ExampleTestCase.h>(
        "ExampleTest", &ExampleTestCase:ExampleTest));

    return testSuite;
}

typedef example;

void ExampleTestCase::ExampleTest()
{

    .......
    assertCondition(medium.Material.size() == 1);
    .......
}
```

Refactoring

MS Test

```
#include statements

#include <afx.h>
using namespace System;
using namespace System::Text;
using namespace Sustem::Collection::Generic;
using namespace Microsoft::VisualStudio::TestTools::UnitTesting

static constant example;

namespace ExampleUnitTest
{
    [TestClass]
    public ref class UnitTest1
    {
        private:
        TestContext^ testContextInstance;

        public:
            property Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ TestContext
            {
                Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ get()
                {
                    return testContextInstance;
                }
                System::Void set(Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ value)
                {
                    testContextInstance = value;
                }
            };

        typedef example;

        [TestMethod]
        void ExampleTest()
        {
            .......
            Assert::IsTrue(medium.Material.size() == 1);
            .......
        }
    };
}
```


MIGRATION OF TEST SUITES FROM PROPRIETARY TO GOOGLE TEST

```
public static Writable<IASTNode> refactorTestCase(Writable<IASTNode> writable, String... args){
    var decl = ASTFactory.getDeclaration("$type* $namespace::suite() {$$statements;}");
    var filename = writable.getNode().getContainingFilename();
    var name = filename.isBlank() ? "MS"
            : Path.of(filename).getParent().getParent().getFileName().toString();
    var match = ASTFinder.findFirst(writable.getNode(), decl);
    if (match==null) {
        return writable;
    }

    writable = replaceAssertCondition(writable);

    writable = replaceAssert(writable);

    var functionsToMove = functionsToMove(writable);

    var othersToMove = othersToMove(writable, decl);

    var moveOthersAsString = toMoveOthersAsString(othersToMove);

    var moveFunctionsAsString = toMoveFunctionsAsString(writable, functionsToMove);

    writable = removeFunctions(writable, functionsToMove);
    writable = removeOthers(writable, othersToMove);

    match = ASTFinder.findFirst(writable.getNode(), decl);
```

```
 String replaceString = """
namespace %sUnitTest
{
    [TestClass]
    public ref class UnitTest1
    {
        private:
            TestContext^ testContextInstance;

        public:
            property Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ TestContext
            {
                Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ get()
                {
                    return testContextInstance;
                }
                System::Void set(Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ value)
                {
                    testContextInstance = value;
                }
            };
%s
%s
    };
}
 """.formatted(name, indent(moveOthersAsString, 3*4), indent(moveFunctionsAsString, 3*4));
 writable.replace(match.getNodes()[0], replaceString);

 writable = addsIncludesAndNamespaces(writable);

 writable = removeUnusedIncludes(writable);

 return writable.commit();
```


SOLVING CODE DEFECTS & SMELLS

Rule:

```
replace ` strcpy($dest, $src); `with` strcpy_s($dest, sizeof($dest), $src); `
```

Diff:

```diff
51                                                  53
52  -     strcpy(value_ptr, "4022.000.000.0");      54  +     strcpy_s(value_ptr, sizeof(value_ptr), "4022.000.000.0");
53        *valid_ptr = FALSE;                       55        *valid_ptr = FALSE;
```

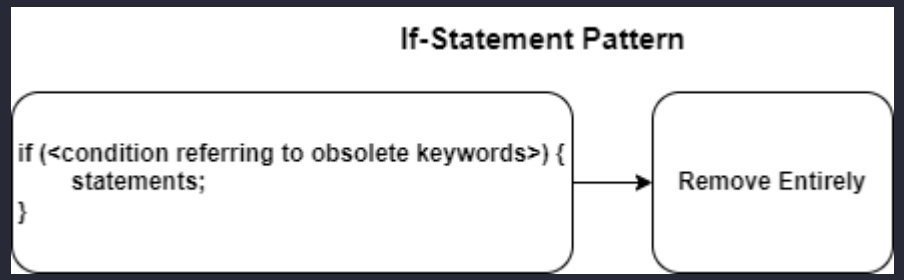
Violation code	Description
6.5.2.2.a	ANSI C functions shall not be called if there is an equivalent safer function
6.10.8.b	Only the predefined macro names __FILE__, __LINE__ and __func__ may be used.
6.7.8.c	For a struct the initializer should be {0} or all fields should be initialized.
5.1.1.1.c	Naming convention of the include guard of an include file named CCBB_foo.h is of format CCBB_FOO_H.
6.10.3.d	Macros shouldn't end with a semicolon.
6.10.3.e	Use a semicolon at the end of a function-like macro call.

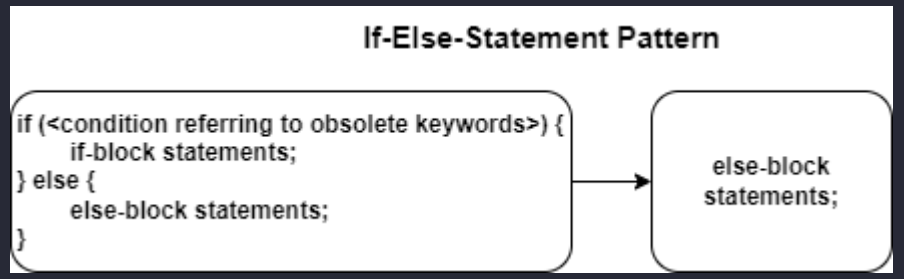
Some violations are related and therefore non-trivial!

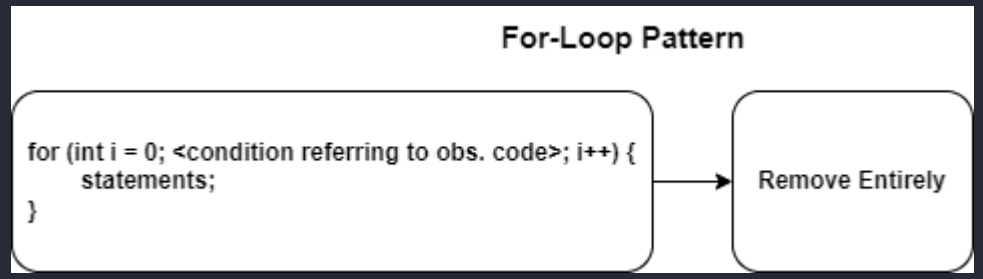
REMOVAL OF OBSOLETE CODE

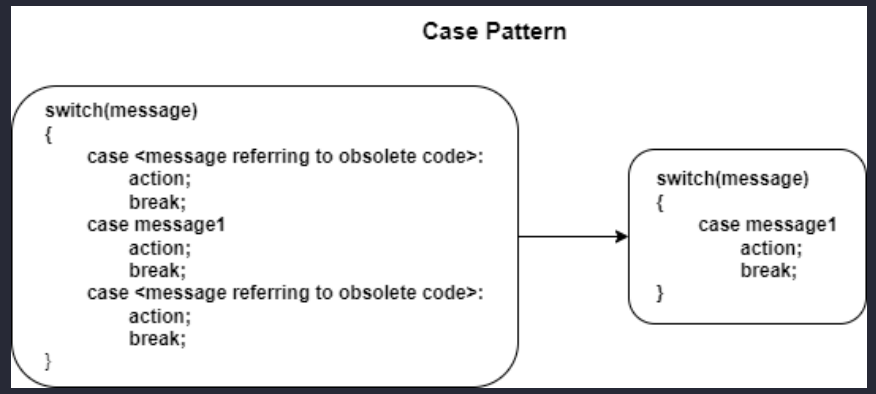
Iterative approach, based on investigation of 4 patterns:

1. Identify common patterns referring to obsolete constructs
2. Remove the patterns
3. Analyze new code state
4. Remove dead code









REMOVAL OF OBSOLETE CODE

```java
public static Writable<IASTNode> removeForLoop(Writable<IASTNode> rewriter, String pattern) {
    NodeUtil.findInFileAsStream(rewriter.getNode(), IASTForStatement.class) //findInFileAsStream only accepts classes
            .filter(f->eligible(f.getConditionExpression(),pattern))
            .forEach(rewriter::remove);
    return rewriter.commit();
}

public static Writable<IASTNode> removeCaseStatement(Writable<IASTNode> rewriter, String pattern) {
    var statements = ASTFactory.getStatements("case $cond: $$statements; break;");
    for (var match : ASTFinder.findStatements(rewriter.getNode(), statements)) {
        if (match.getSingleAsRawSignature("$cond").matches(pattern)) {
            rewriter.remove(match.getNodes());
        }
    }
    return rewriter.commit();
}
```

```java
public static Writable<IASTNode> removeIfStatement(Writable<IASTNode> rewriter, String pattern) {
    NodeUtil.findInFileAsStream(rewriter.getNode(), IASTIfStatement.class)
        .filter(is->eligible(is.getConditionExpression(), pattern))
        .forEach(is->{
            if(is.getElseClause()==null) {
                rewriter.remove(is); //if will never be trigger anymore
            }
            else {
                var ec = is.getElseClause();
                var hasDeclaration = ASTFinder.findFirst( ec, IASTDeclarationStatement.class)!=null;
                if( hasDeclaration) {
                    rewriter.replace(is, ec.getRawSignature());
                }
                else {
                    var elseBlock = is.getElseClause().getRawSignature().trim();
                    if (elseBlock.charAt(0) == '{' && elseBlock.charAt(elseBlock.length() - 1) == '}') {
                        var replacement = elseBlock.substring(1, elseBlock.length() - 1);
                        rewriter.replace(is, replacement.trim().indent(-4));
                    } else {
                        rewriter.replace(is, elseBlock);
                    }
                }
            }
        });
    return rewriter.commit();
}
```

REMOVAL OF OBSOLETE CODE

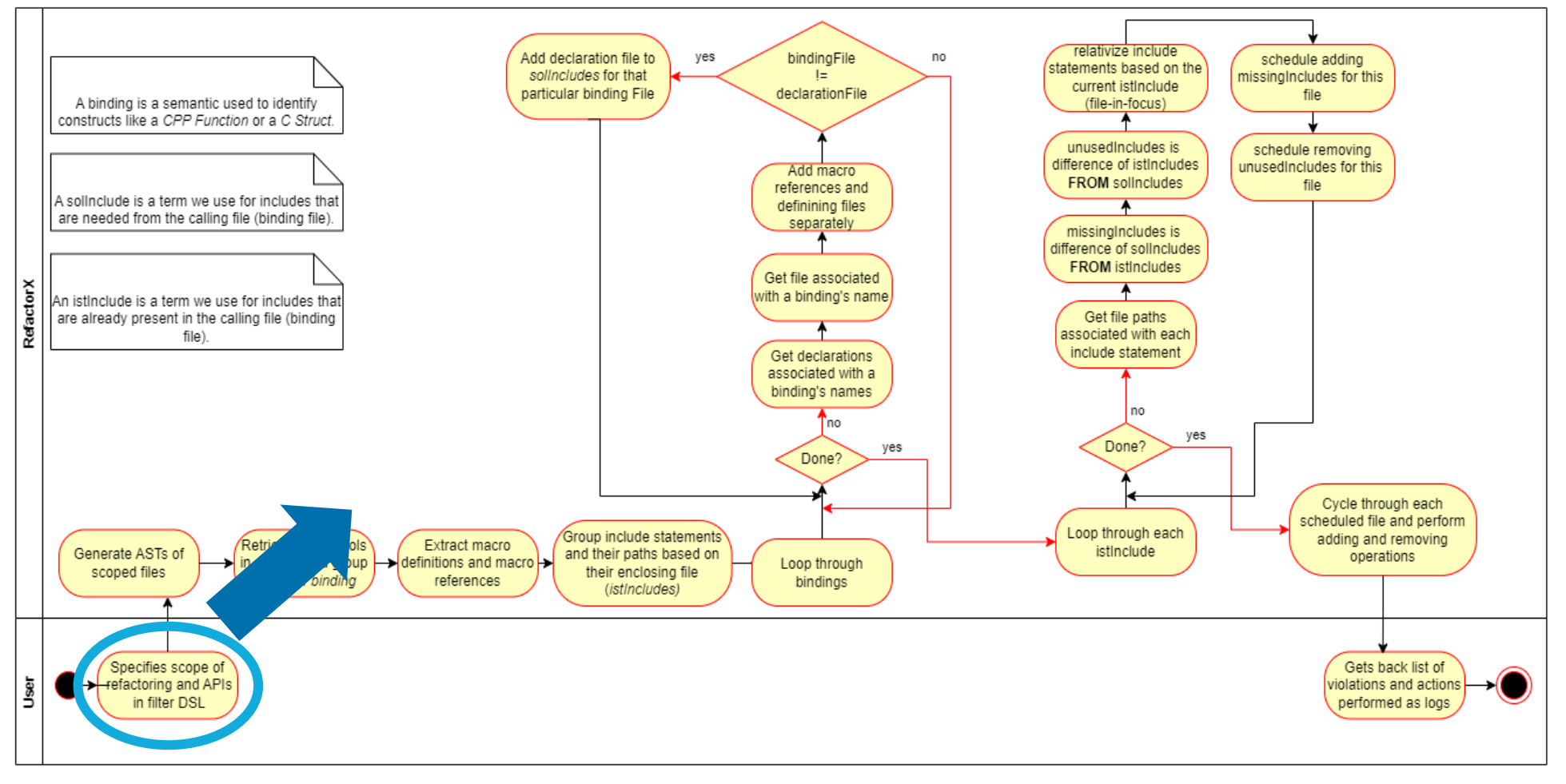
```
switch(message)
{
case tatramed::data::object::C_DO_Plan::Added_Beam:
    m_Plan->OnAddedBeam((tatramed::data::object::C_DO_Beam*)para
    break;
case tatramed::data::object::C_DO_Plan::Deleted_Beam:
    m_Plan->OnDeletedBeam((tatramed::data::object::C_DO_Beam*)pa
    break;
case tatramed::data::object::C_DO_Plan::Changed_Name:
    m_Plan->OnChangedName();
    break;
case tatramed::data::object::C_DO_Plan::Changed_Notes:
    m_Plan->OnChangedNotes();
    break;
case tatramed::data::object::C_DO_Plan::Changed_Intent:
    m_Plan->OnChangedPlanIntent();
    break;
}
```

```
27      switch(message)
28      {
29  +
30  +
31      case tatramed::data::object::C_DO_Plan::Changed_Name:
32          m_Plan->OnChangedName();
33          break;
34      case tatramed::data::object::C_DO_Plan::Changed_Notes:
35          m_Plan->OnChangedNotes();
36          break;
37      case tatramed::data::object::C_DO_Plan::Changed_Intent:
38          m_Plan->OnChangedPlanIntent();
39          break;
40      }
```

```
for(unsigned i = 0; i < pBlock->m_Bars; i++)
{
    ids[i] = (int)(INT_PTR)pInfo->m_arrBarID[i];
}
```

```
if(m_bCreateBeamInProgress)
{
    ASSERT(m_CreatedBeam == nullptr);
    m_CreatedBeam = doBeam;
    m_bCreateBeamInProgress = false;
}
```


MAKE SOURCE FILES SELF-CONTAINED



Implemented in ~600 Java LoC

Using Renaissance + Eclipse CDT APIs

Standard + company specific rules

CONCLUSIONS

Automated code refactoring technically feasible

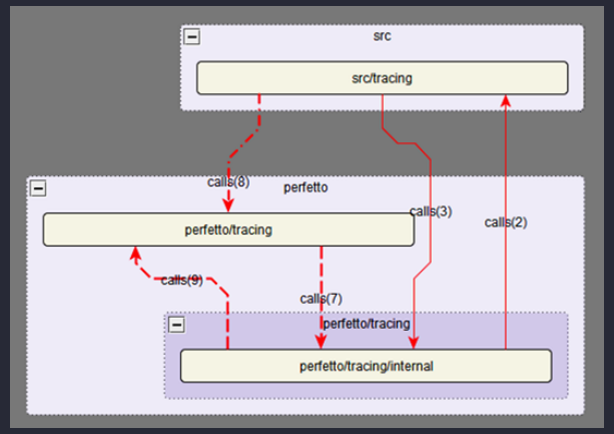
Possible when there’s a deterministic rule for replacing matched pattern

Notes of warning

- Always test code refactoring before unleashing into the wild
- Scale needs to be sufficiently big to achieve ROI

DO WE STILL HAVE TIME?

ARCHITECTURE ANALYSIS WITH RENAISSANCE



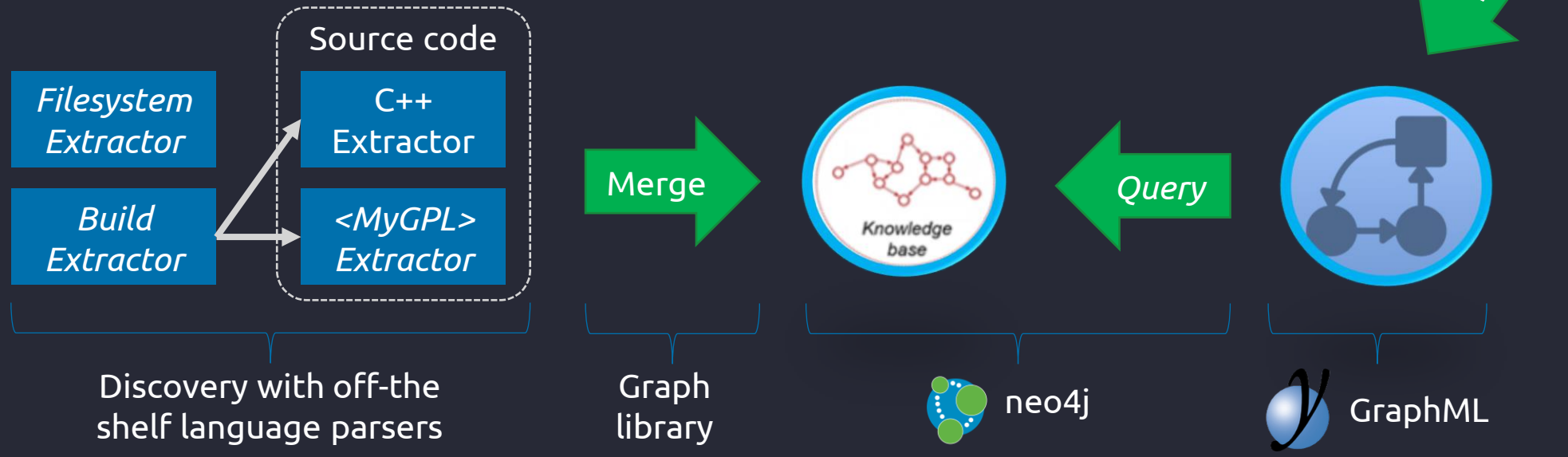
Architecture Analysis

- Visualize code depending on COM IDL
- Report IPC events/calls between processes
- Visualize cyclic dependencies

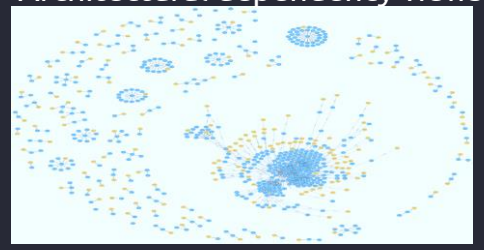
MODEL-BASED KNOWLEDGE EXTRACTION & ANALYSIS

Tailored code analysis and visualization

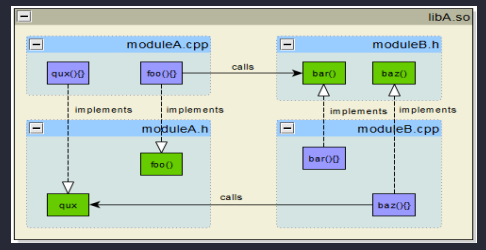
- Extensible to support technology mixture
- Customizable to answer analysis question
- Performant to handle large-scale code base



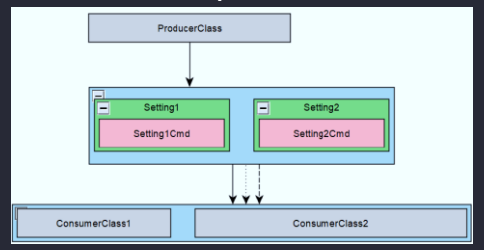
Architectural dependency views



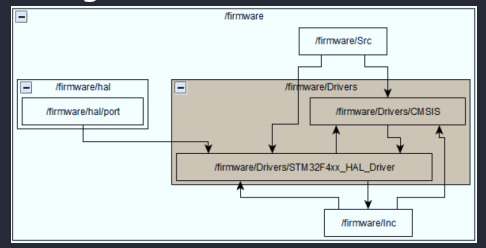
Architectural structure views



Architectural pattern views



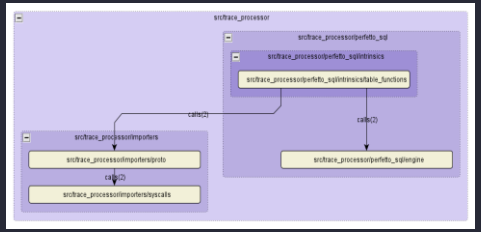
Design views



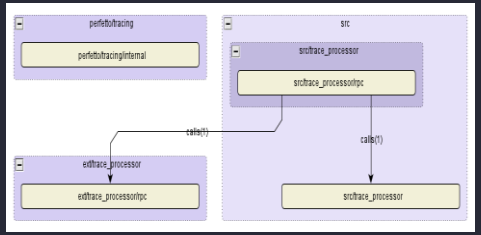
DEMO: MODEL-BASED ANALYSIS USING REBORN

1. Extract code and publish to Knowledge Base
2. Extract project information and refine Knowledge Base
3. Generate views
4. View generated views in YEd

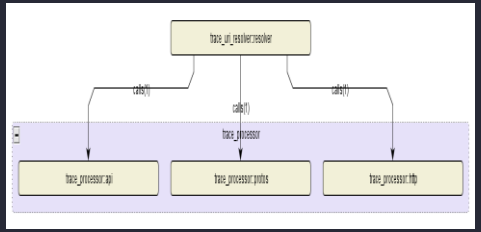
Folder dependencies view



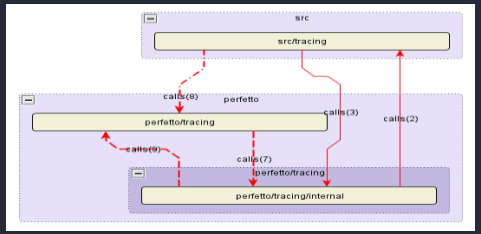
Project dependencies view

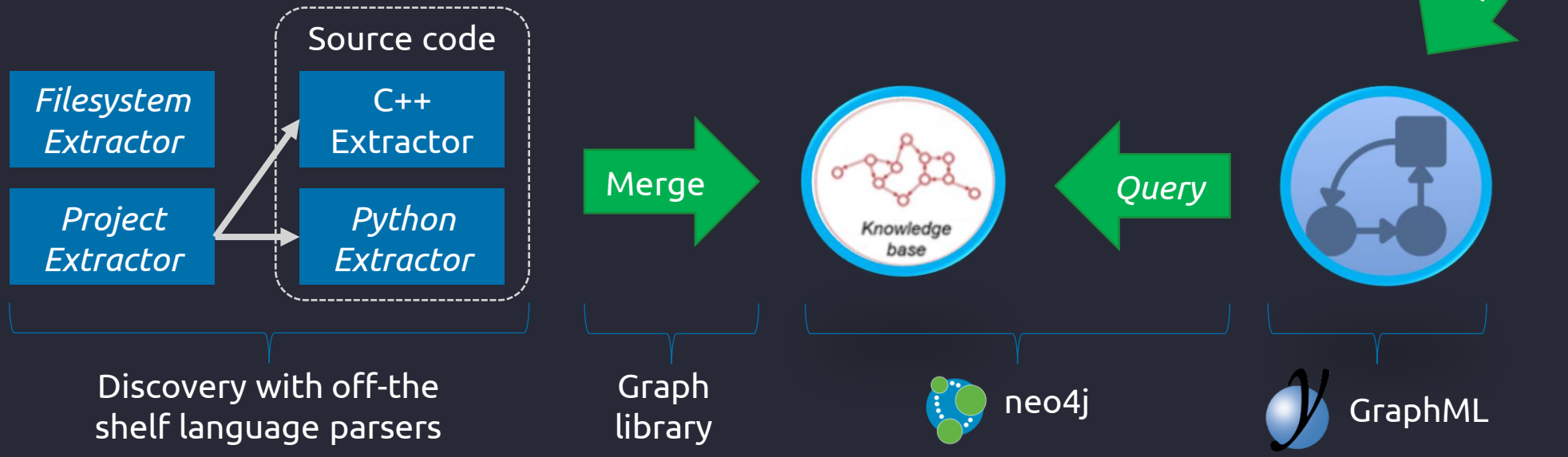


Namespaces dependencies view



Cyclic dependencies view





KEY TAKE AWAY

Keep your software forever young by automating software maintenance!

About Capgemini Engineering

Capgemini Engineering combines, under one brand, a unique set of strengths from across the Capgemini Group: the world leading engineering and R&D services of Altran – acquired by Capgemini in 2020 - and Capgemini's digital manufacturing expertise. With broad industry knowledge and cutting-edge technologies in digital and software, Capgemini Engineering supports the convergence of the physical and digital worlds. Combined with the capabilities of the rest of the Group, it helps clients to accelerate their journey towards Intelligent Industry. Capgemini Engineering has more than 52,000 engineer and scientist team members in over 30 countries across sectors including aeronautics, automotive, railways, communications, energy, life sciences, semiconductors, software & internet, space & defense, and consumer products.

Capgemini Engineering is an integral part of the Capgemini Group, a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided everyday by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of over 325,000 team members more than 50 countries. With its strong 55-year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering and platforms. The Group reported in 2021 global revenues of €18 billion.

Get the Future You Want | www.capgemini.com/capgemini-engineering