Mining and Analyzing Source Code Changes

Martin Pinzger & Veit Frick
Software Engineering Research Group
Universität Klagenfurt, Austria
http://serg.aau.at

My research goals

Build the next generation of software development tools and online collaboration platforms

More info
https://pinzger.github.io/

My research goals

Build the next generation of software development tools and online collaboration platforms

Helping developers to understand changes and their impact

More info
https://pinzger.github.io/

My research goals

Build the next generation of software development tools and online collaboration platforms

Helping developers to understand changes and their impact

Improve evaluation and validation in software engineering

More info
https://pinzger.github.io/

My research goals

Build the next generation of software development tools and online collaboration platforms

Helping developers to understand changes and their impact

Automating software engineering tasks

Improve evaluation and validation in software engineering

More info
https://pinzger.github.io/

My research goals

Build the next generation of software development tools and online collaboration platforms

Helping developers to understand changes and their impact

Automating software engineering tasks

Improve evaluation and validation in software engineering

Overview

The need for understanding code changes

Fine-grained source code change extraction

- ChangeDistiller and IJM

Hands on IJM and DiffViz

Lehman’s Law of Software Evolution

Continuing change

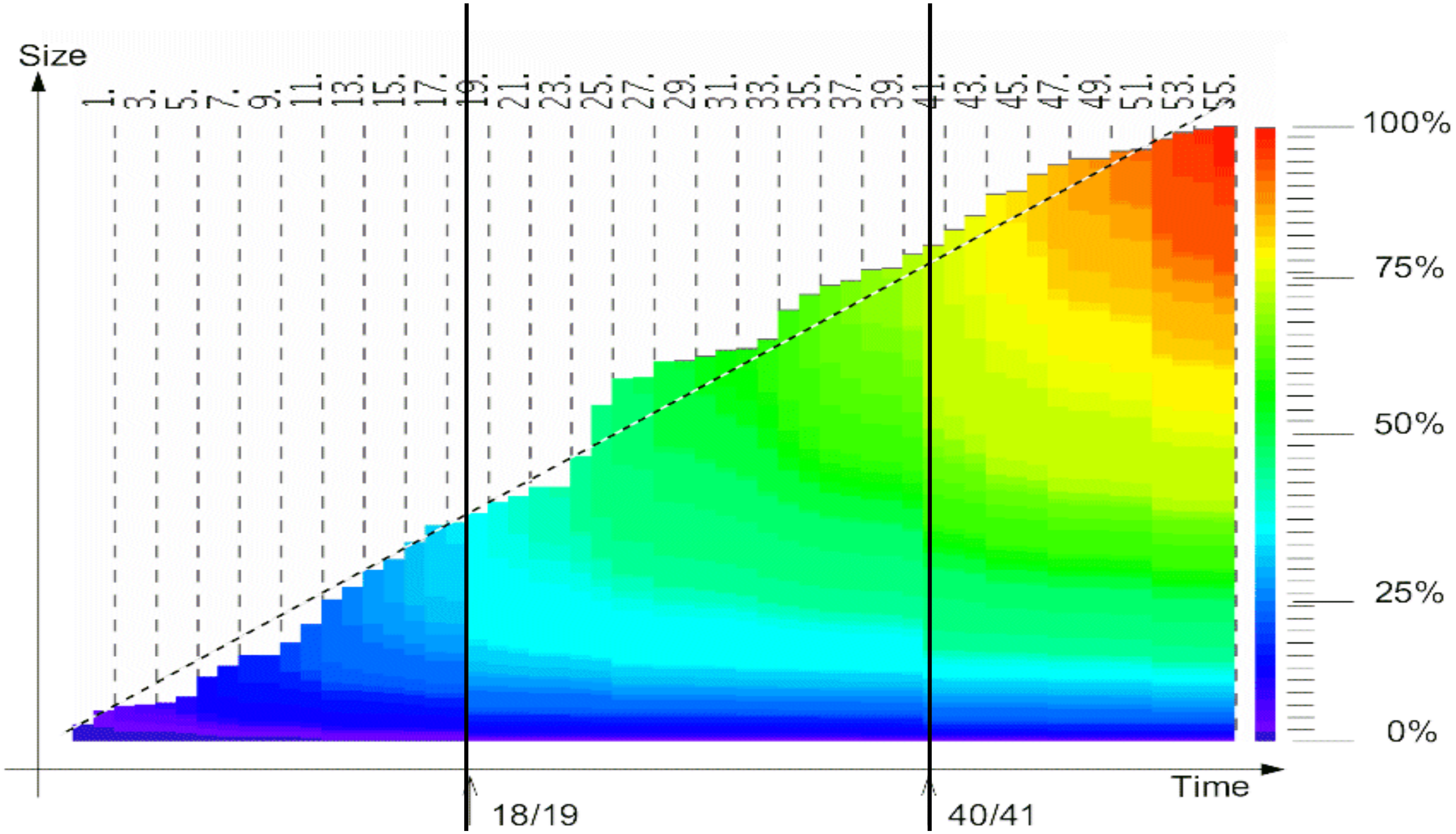
A program that is used in a real-world environment must change, or become progressively less useful in that environment.

Increasing complexity

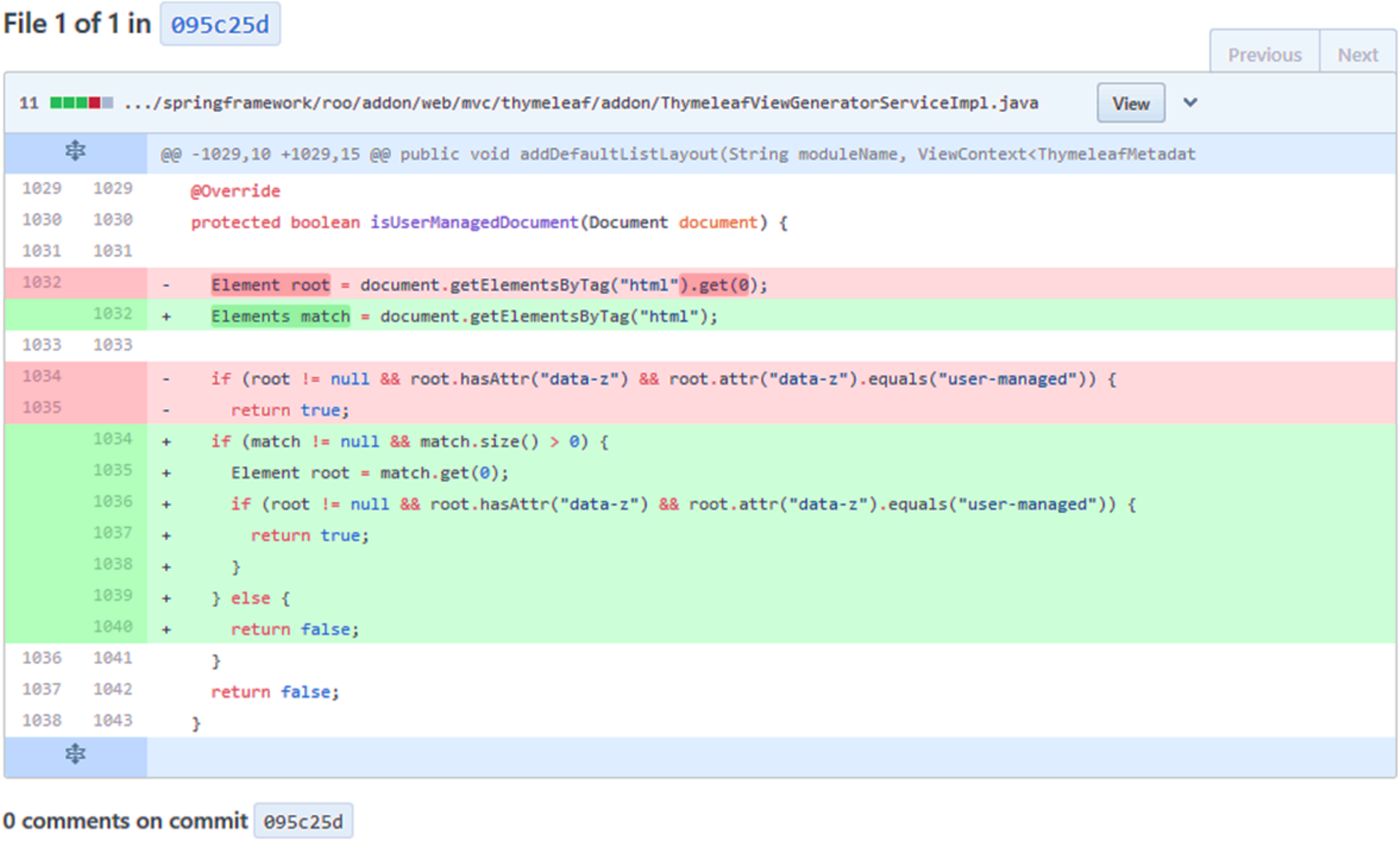
As a program evolves, it becomes more complex, and extra resources are needed to preserve and simplify its structure.

For more information read Lehman and Belady, 1985

Lehman's Laws in Mozilla



What did change?



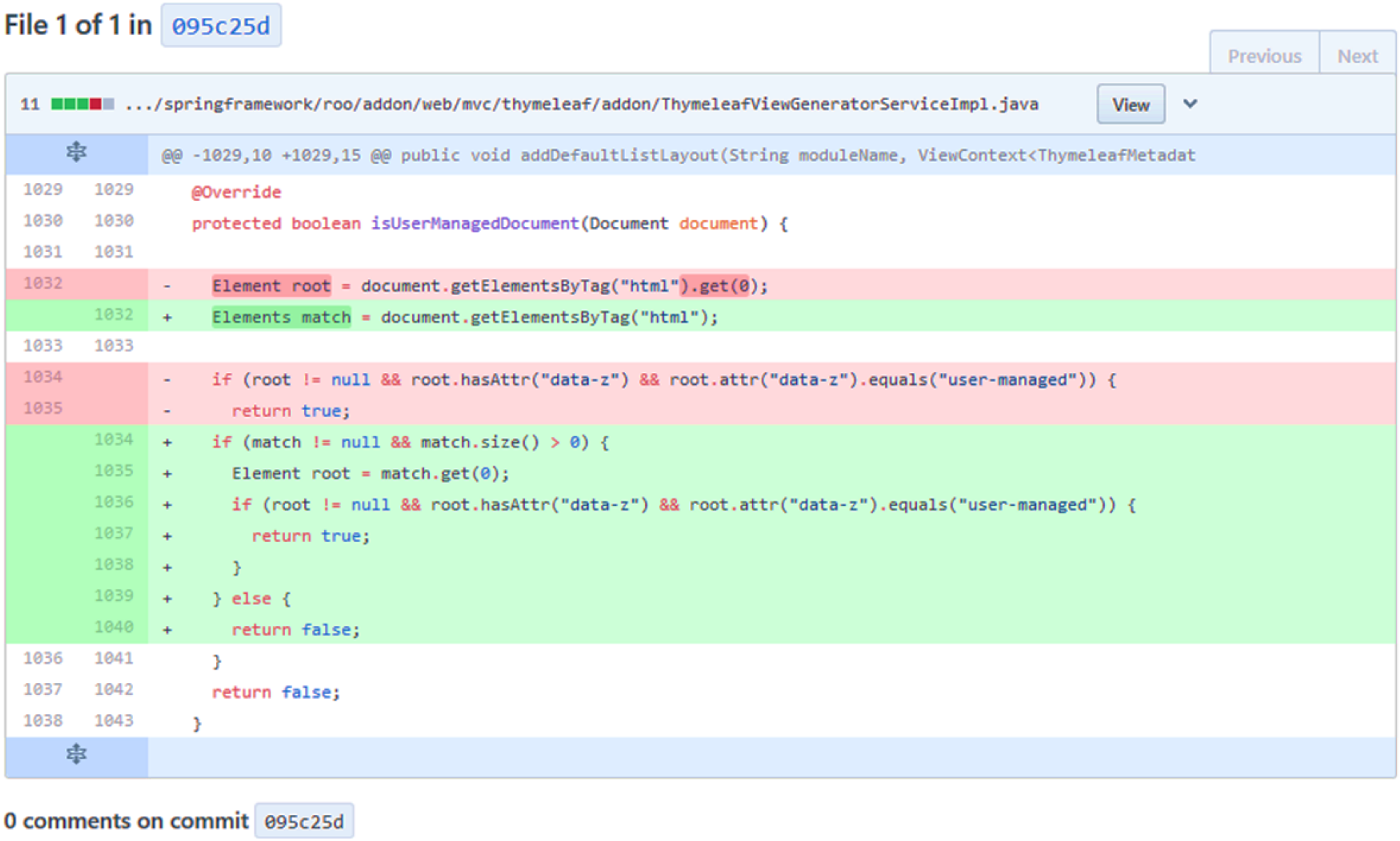
File 1 of 1 in 095c25d

11 🟩🟩🟩🟥▨ .../springframework/roo/addon/web/mvc/thymeleaf/addon/ThymeleafViewGeneratorServiceImpl.java

```
@@ -1029,10 +1029,15 @@ public void addDefaultListLayout(String moduleName, ViewContext<ThymeleafMetadat
   @Override
   protected boolean isUserManagedDocument(Document document) {

-    Element root = document.getElementsByTag("html").get(0);
+    Elements match = document.getElementsByTag("html");

-    if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
-      return true;
+    if (match != null && match.size() > 0) {
+      Element root = match.get(0);
+      if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
+        return true;
+      }
+    } else {
+      return false;
     }
     return false;
   }
```

0 comments on commit 095c25d

What is the change impact?



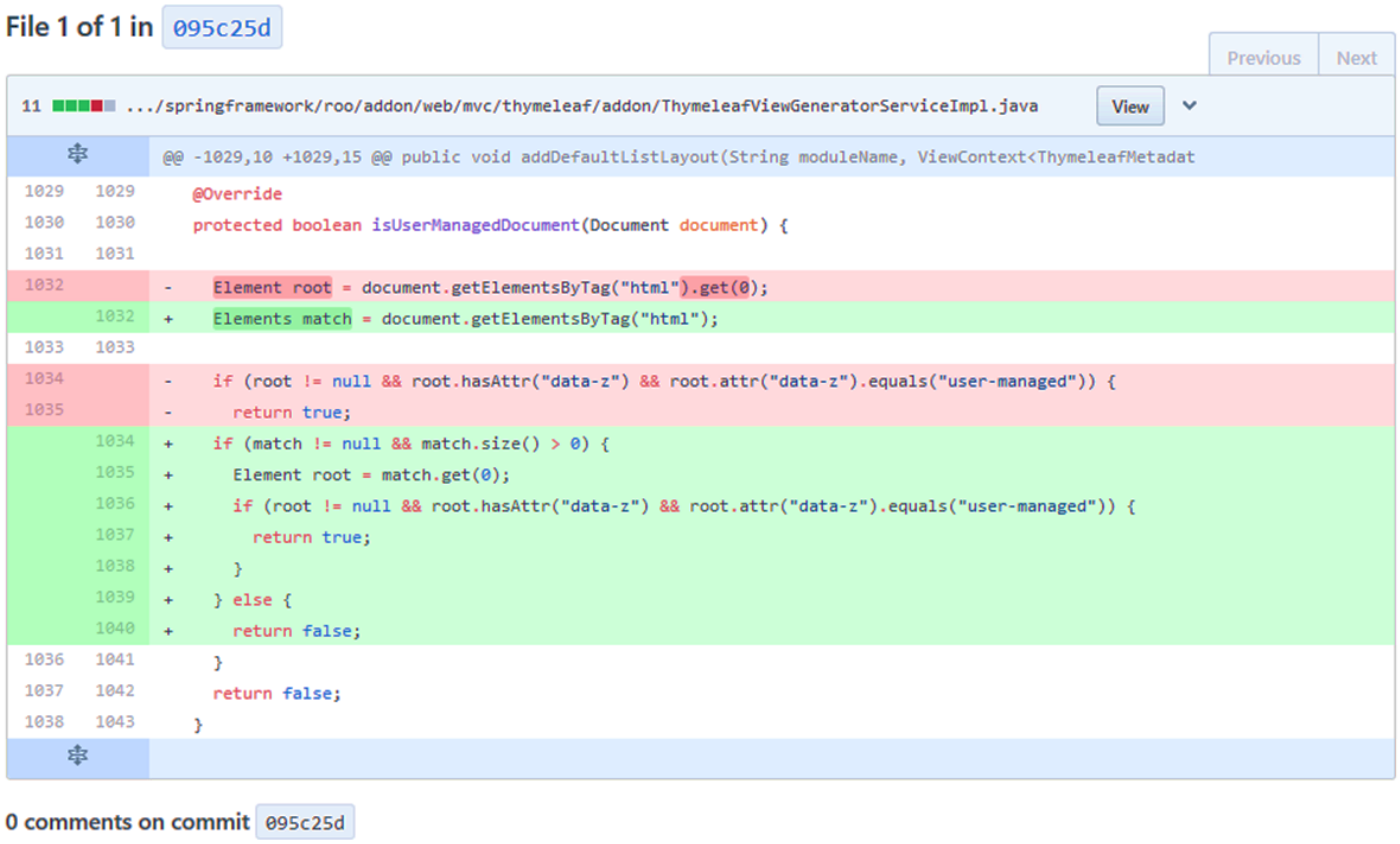
File 1 of 1 in 095c25d

.../springframework/roo/addon/web/mvc/thymeleaf/addon/ThymeleafViewGeneratorServiceImpl.java

```
@@ -1029,10 +1029,15 @@ public void addDefaultListLayout(String moduleName, ViewContext<ThymeleafMetadat
   @Override
   protected boolean isUserManagedDocument(Document document) {

-    Element root = document.getElementsByTag("html").get(0);
+    Elements match = document.getElementsByTag("html");

-    if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
-      return true;
+    if (match != null && match.size() > 0) {
+      Element root = match.get(0);
+      if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
+        return true;
+      }
+    } else {
+      return false;
     }
     return false;
   }
```

0 comments on commit 095c25d

Do the changes affect my code?



File 1 of 1 in 095c25d

11 🟩🟩🟩🟥 .../springframework/roo/addon/web/mvc/thymeleaf/addon/ThymeleafViewGeneratorServiceImpl.java

```
@@ -1029,10 +1029,15 @@ public void addDefaultListLayout(String moduleName, ViewContext<ThymeleafMetadat
  @Override
  protected boolean isUserManagedDocument(Document document) {

-  Element root = document.getElementsByTag("html").get(0);
+  Elements match = document.getElementsByTag("html");

-  if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
-    return true;
+  if (match != null && match.size() > 0) {
+    Element root = match.get(0);
+    if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
+      return true;
+    }
+  } else {
+    return false;
   }
   return false;
  }
```

0 comments on commit 095c25d

Understanding changes and their impact

Existing tools lack support for comprehending changes

> "How do software engineers understand code changes? - an exploratory study in industry", Tao et al. 2012

Developers need to reconstruct the detailed context and impact of each change which is time consuming and error prone

> "An exploratory study of awareness interests about software modifications", Kim 2011

We need better support to analyze and comprehend changes and their impact

Overview of our tools

ChangeDistiller and **IJM**: Fine-grained evolution of Java classes

BuildDiff: Evolution of Maven build scripts

WSDLDiff: Evolution of service-oriented systems

FMDiff: Evolution of feature models

ChangeDistiller: tree differencing for fine-grained source code change extraction

Beat Fluri, Michael Würsch, Martin Pinzger, and Harald Gall

Extracting source code changes using ASTs

Using tree differencing, we can determine

```
public void method(D d) {
    if (d != null) {
        d.foo();
        d.bar();
    }
}
```

```
public void method(D d) {
    d.foo();
    d.bar();
}
```

Extracting source code changes using ASTs

Using tree differencing, we can determine

- Enclosing entity (root node)

```
public void method(D d) {
    if (d != null) {
        d.foo();
        d.bar();
    }
}
```

```
public void method(D d) {
    d.foo();
    d.bar();
}
```

Extracting source code changes using ASTs

Using tree differencing, we can determine

- Enclosing entity (root node)
- Kind of statement which changed (node information)

```
public void method(D d) {
    if (d != null) {
        d.foo();
        d.bar();
    }
}
```

```
public void method(D d) {
    d.foo();
    d.bar();
}
```

Extracting source code changes using ASTs

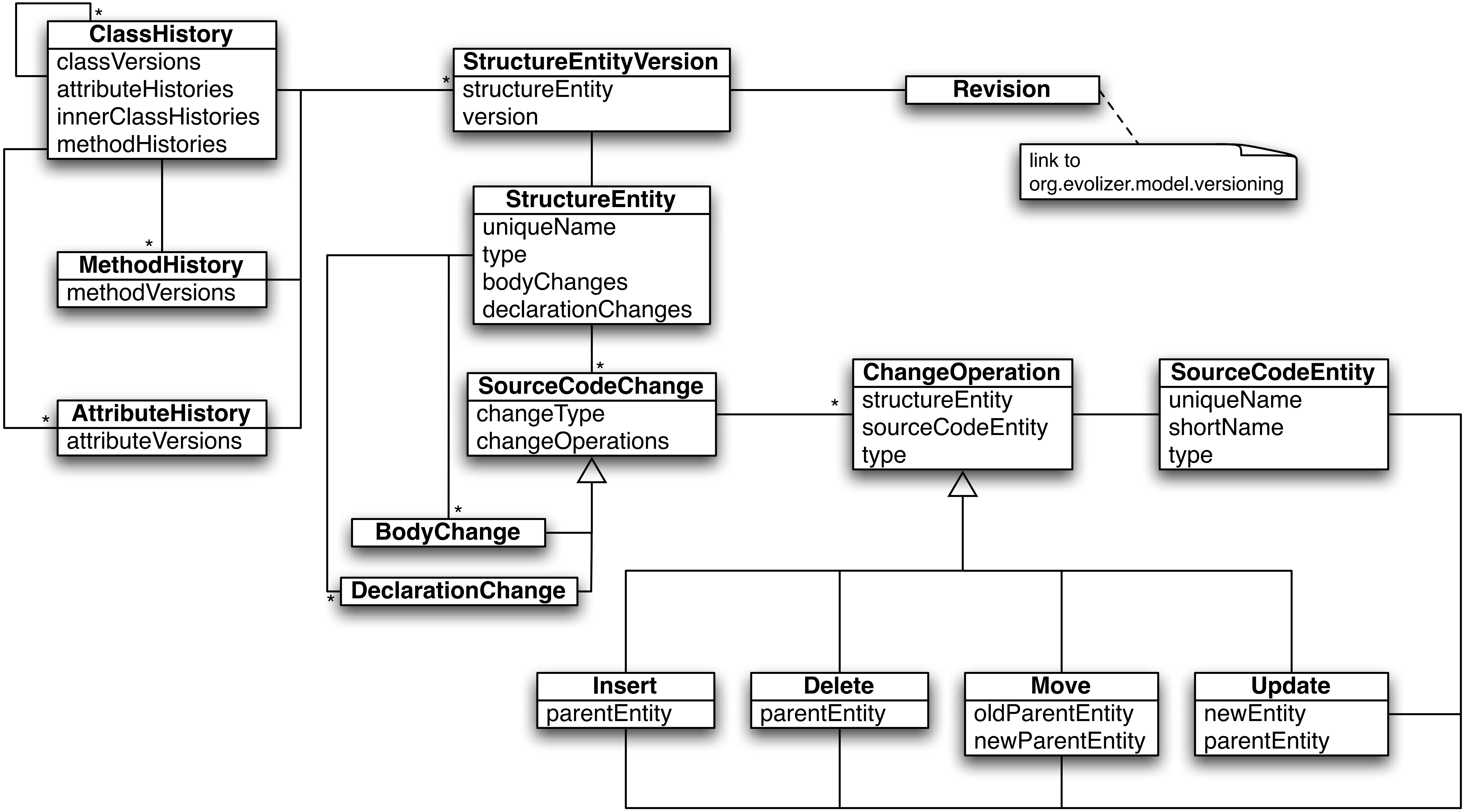
Using tree differencing, we can determine

- Enclosing entity (root node)
- Kind of statement which changed (node information)
- Kind of change (tree edit operation)

```
public void method(D d) {
    if (d != null) {
        d.foo();
        d.bar();
    }
}
```

```
public void method(D d) {
    d.foo();
    d.bar();
}
```

ChangeDistiller model



Change type categories

cDecl = changes to class declarations

oState = insertion and deletion of class attributes

func = insertion and deletion of methods

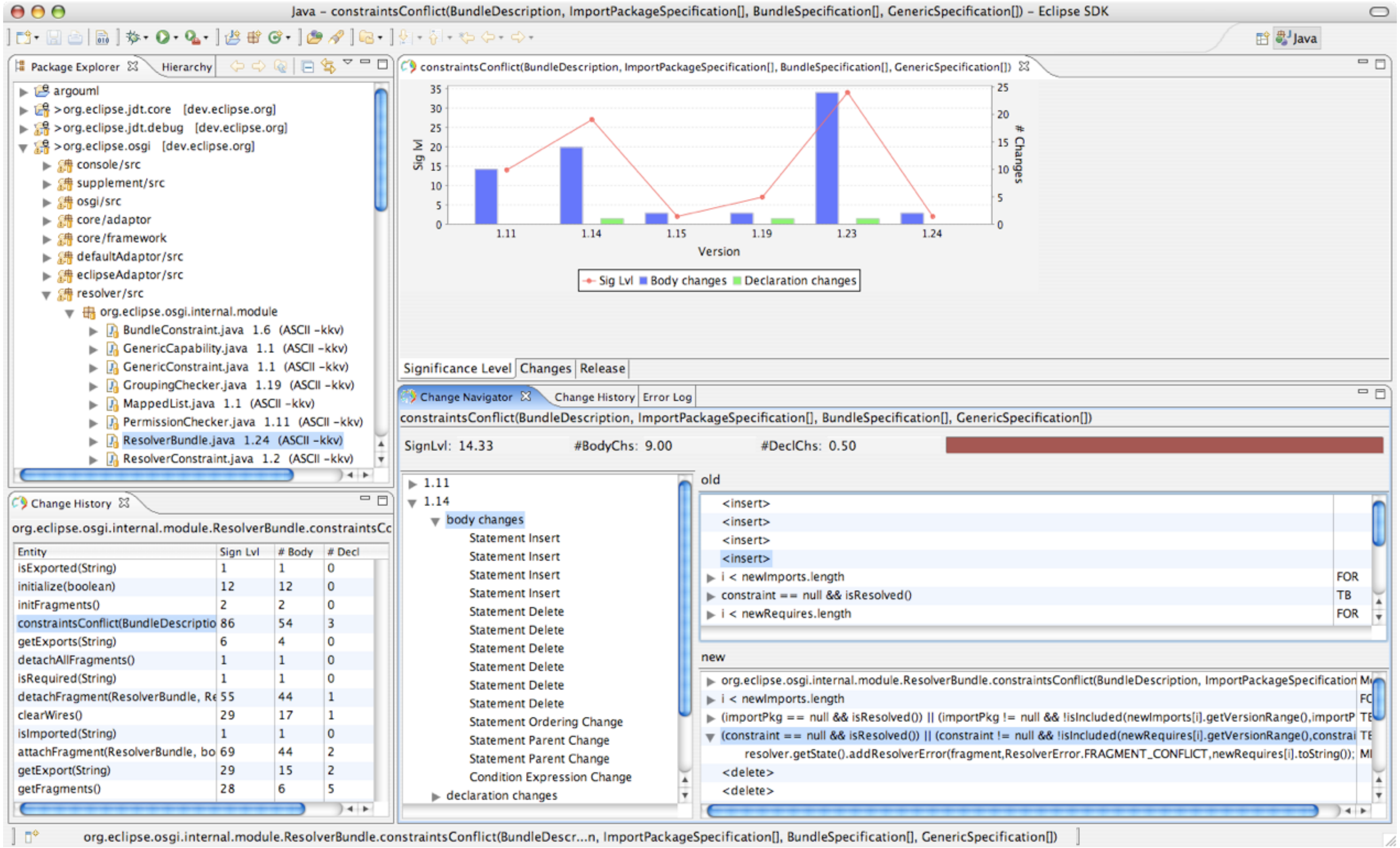
mDecl = changes to method declarations

stmt = insertion, deletion, ordering of executable statements

cond = changes to conditional expressions

else = insertion and deletion of else-parts

ChangeDistiller tool


Java – constraintsConflict(BundleDescription, ImportPackageSpecification[], BundleSpecification[], GenericSpecification[]) – Eclipse SDK
Java
Package Explorer
Hierarchy
constraintsConflict(BundleDescription, ImportPackageSpecification[], BundleSpecification[], GenericSpecification[])
35
30
25
20
15
10
5
0
Sig lvl
25
20
15
10
5
0
# Changes
1.11
1.14
1.15
1.19
1.23
1.24
Version
Sig Lvl
Body changes
Declaration changes
Significance Level
Changes
Release
Change Navigator
Change History
Error Log

argouml
>org.eclipse.jdt.core [dev.eclipse.org]
>org.eclipse.jdt.debug [dev.eclipse.org]
>org.eclipse.osgi [dev.eclipse.org]
console/src
supplement/src
osgi/src
core/adaptor
core/framework
defaultAdaptor/src
eclipseAdaptor/src
resolver/src
org.eclipse.osgi.internal.module
BundleConstraint.java 1.6 (ASCII -kkv)
GenericCapability.java 1.1 (ASCII -kkv)
GenericConstraint.java 1.1 (ASCII -kkv)
GroupingChecker.java 1.19 (ASCII -kkv)
MappedList.java 1.1 (ASCII -kkv)
PermissionChecker.java 1.11 (ASCII -kkv)
ResolverBundle.java 1.24 (ASCII -kkv)
ResolverConstraint.java 1.2 (ASCII -kkv)

Change History

org.eclipse.osgi.internal.module.ResolverBundle.constraintsCc

Entity	Sign Lvl	# Body	# Decl
isExported(String)	1	1	0
initialize(boolean)	12	12	0
initFragments()	2	2	0
constraintsConflict(BundleDescriptio	86	54	3
getExports(String)	6	4	0
detachAllFragments()	1	1	0
isRequired(String)	1	1	0
detachFragment(ResolverBundle, Re	55	44	1
clearWires()	29	17	1
isImported(String)	1	1	0
attachFragment(ResolverBundle, bo	69	44	2
getExport(String)	29	15	2
getFragments()	28	6	5

constraintsConflict(BundleDescription, ImportPackageSpecification[], BundleSpecification[], GenericSpecification[])

SignLvl: 14.33 #BodyChs: 9.00 #DeclChs: 0.50

1.11
1.14
body changes
Statement Insert
Statement Insert
Statement Insert
Statement Insert
Statement Delete
Statement Delete
Statement Delete
Statement Delete
Statement Delete
Statement Delete
Statement Ordering Change
Statement Parent Change
Statement Parent Change
Condition Expression Change
declaration changes

old

```
<insert>
<insert>
<insert>
<insert>
i < newImports.length                                   FOR
constraint == null && isResolved()                      TB
i < newRequires.length                                  FOR
```

new

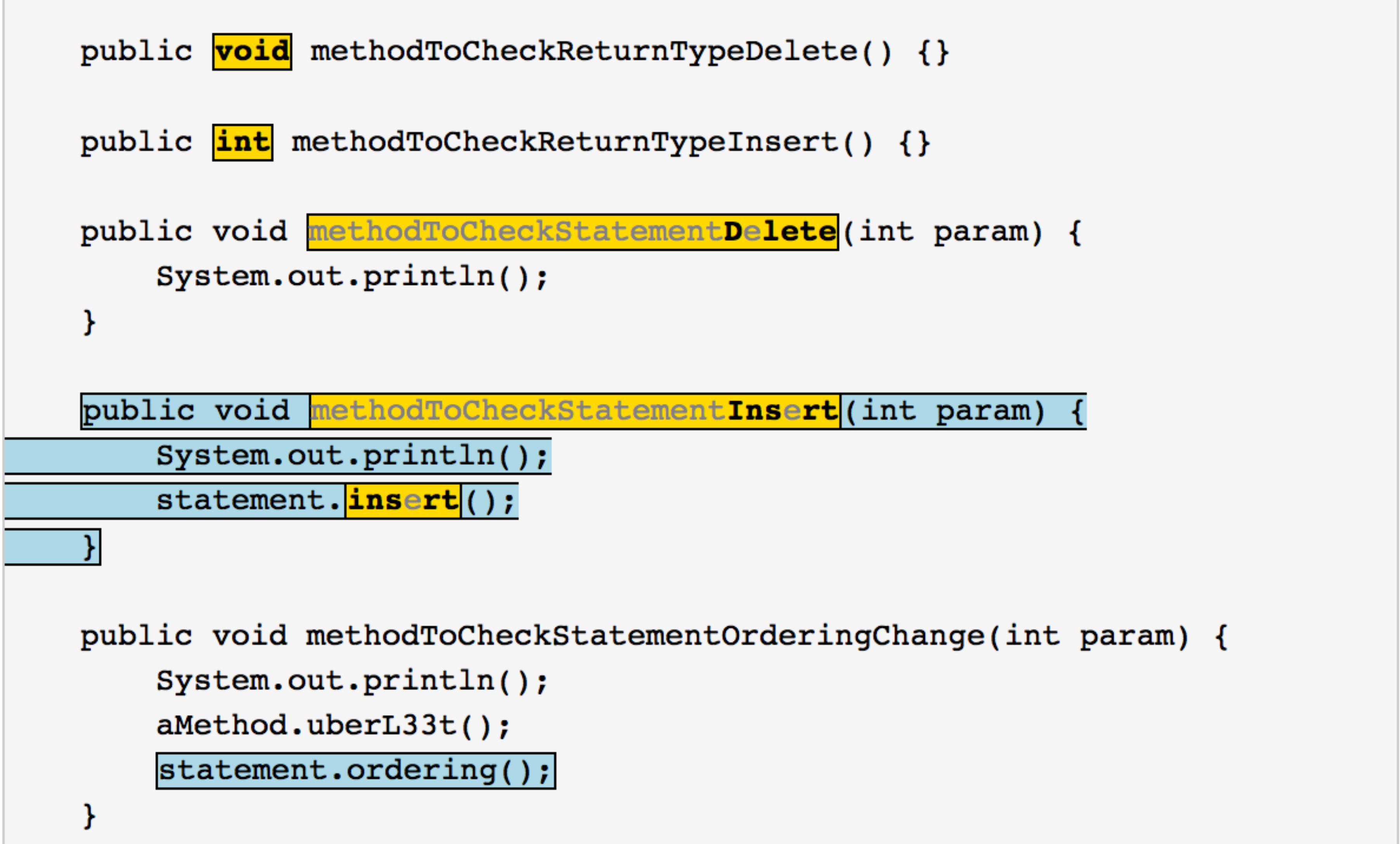
```
org.eclipse.osgi.internal.module.ResolverBundle.constraintsConflict(BundleDescription, ImportPackageSpecification Mı
i < newImports.length                                                                                     FC
(importPkg == null && isResolved()) || (importPkg != null && !isIncluded(newImports[i].getVersionRange(),importP TE
(constraint == null && isResolved()) || (constraint != null && !isIncluded(newRequires[i].getVersionRange(),constrai TE
    resolver.getState().addResolverError(fragment,ResolverError.FRAGMENT_CONFLICT,newRequires[i].toString()); MI
<delete>
<delete>
```

org.eclipse.osgi.internal.module.ResolverBundle.constraintsConflict(BundleDescr...n, ImportPackageSpecification[], BundleSpecification[], GenericSpecification[])

https://bitbucket.org/sealuzh/tools-changedistiller/wiki/Home

GumTree

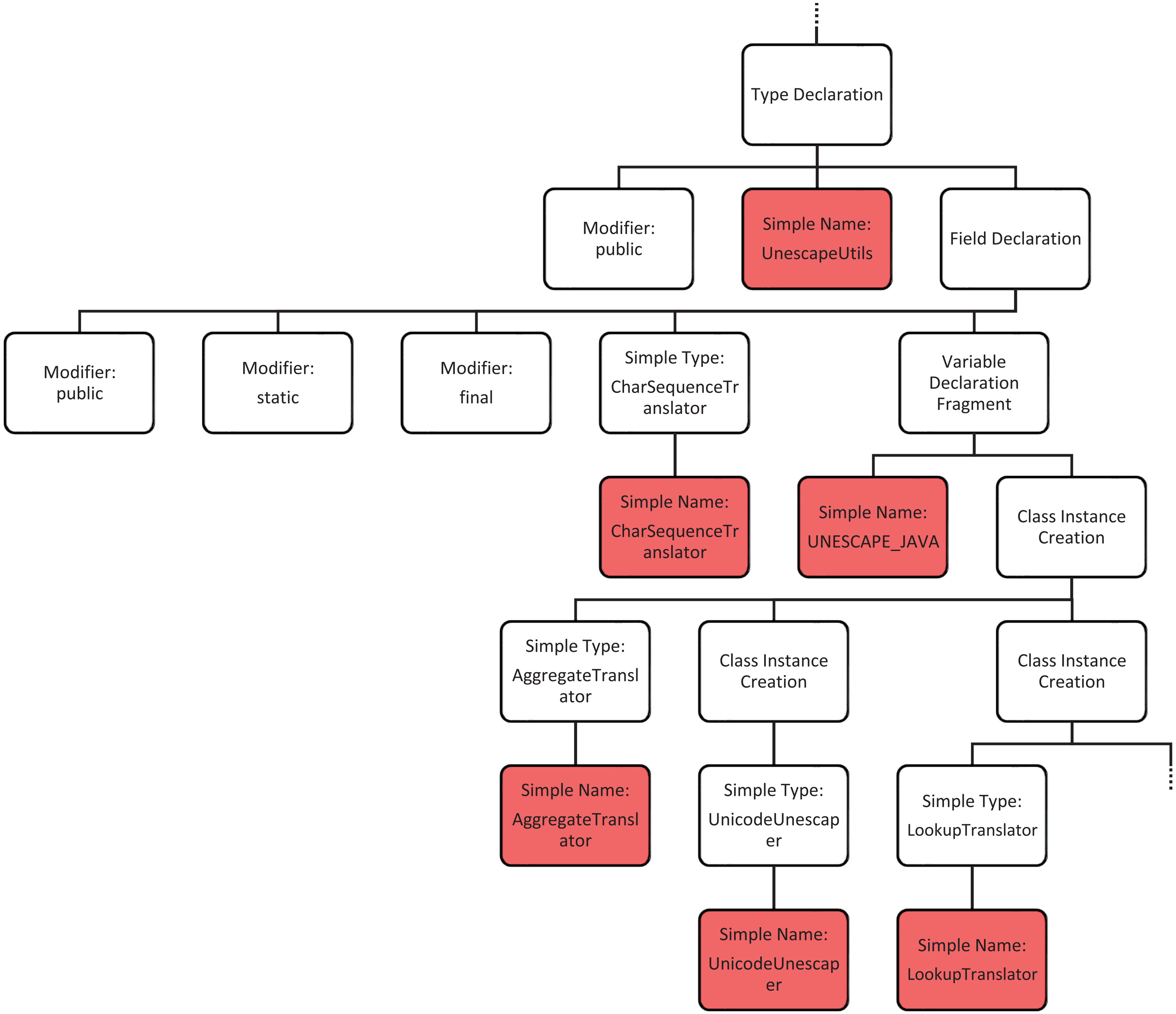
Improvements over ChangeDistiller



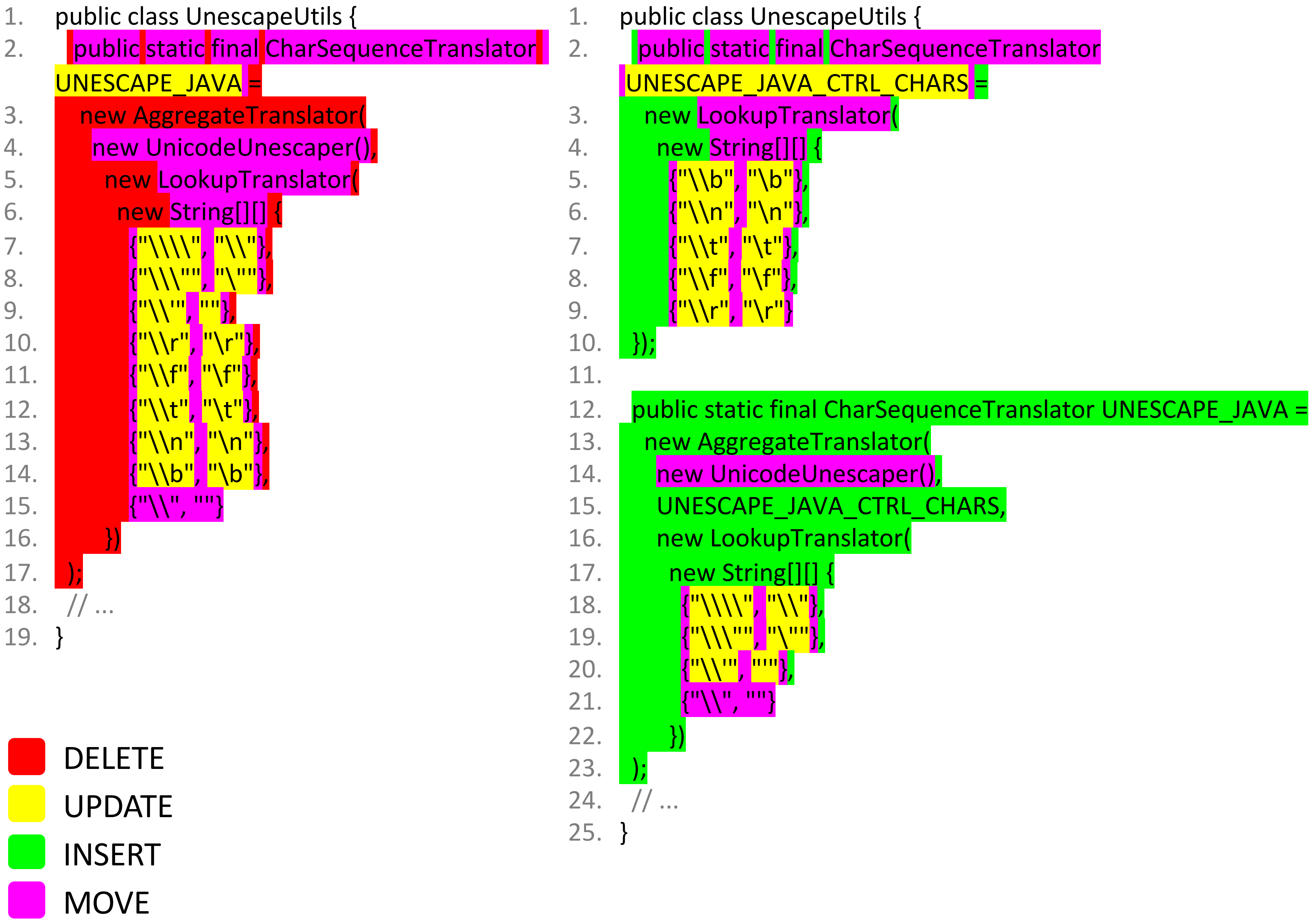
```
    public void methodToCheckReturnTypeDelete() {}

    public int methodToCheckReturnTypeInsert() {}

    public void methodToCheckStatementDelete(int param) {
        System.out.println();
    }

    public void methodToCheckStatementInsert(int param) {
        System.out.println();
        statement.insert();
    }

    public void methodToCheckStatementOrderingChange(int param) {
        System.out.println();
        aMethod.uberL33t();
        statement.ordering();
    }
```

“Fine-grained and accurate source code differencing”, Falleri et al. 2014

GumTree AST used for diffing


Type Declaration
Modifier: public
Simple Name: UnescapeUtils
Field Declaration
Modifier: public
Modifier: static
Modifier: final
Simple Type: CharSequenceTranslator
Variable Declaration Fragment
Simple Name: CharSequenceTranslator
Simple Name: UNESCAPE_JAVA
Class Instance Creation
Simple Type: AggregateTranslator
Class Instance Creation
Class Instance Creation
Simple Name: AggregateTranslator
Simple Type: UnicodeUnescaper
Simple Type: LookupTranslator
Simple Name: UnicodeUnescaper
Simple Name: LookupTranslator

AST Diff of GumTree



```
public class UnescapeUtils {
  public static final CharSequenceTranslator
UNESCAPE_JAVA =
  new AggregateTranslator(
    new UnicodeUnescaper(),
      new LookupTranslator(
        new String[][] {
          {"\\\\", "\\"},
          {"\\\"", "\""},
          {"\\'", "'"},
          {"\\r", "\r"},
          {"\\f", "\f"},
          {"\\t", "\t"},
          {"\\n", "\n"},
          {"\\b", "\b"},
          {"\\", ""}
      })
  );
  // ...
}
```

```
public class UnescapeUtils {
  public static final CharSequenceTranslator
UNESCAPE_JAVA_CTRL_CHARS =
  new LookupTranslator(
    new String[][] {
      {"\\b", "\b"},
      {"\\n", "\n"},
      {"\\t", "\t"},
      {"\\f", "\f"},
      {"\\r", "\r"}
  });

  public static final CharSequenceTranslator UNESCAPE_JAVA =
  new AggregateTranslator(
    new UnicodeUnescaper(),
    UNESCAPE_JAVA_CTRL_CHARS,
    new LookupTranslator(
      new String[][] {
        {"\\\\", "\\"},
        {"\\\"", "\""},
        {"\\'", "'"},
        {"\\", ""}
    })
  );
  // ...
}
```

🟥 DELETE

🟨 UPDATE

🟩 INSERT

🟪 MOVE

Too many unnecessary edits!

IJM: generating accurate and compact edit scripts using tree differencing

Veit Frick, Thomas Grassauer, Fabian Beck, and Martin Pinzger

IJM

Iterative Java Matcher (IJM)

- Builds upon GumTree

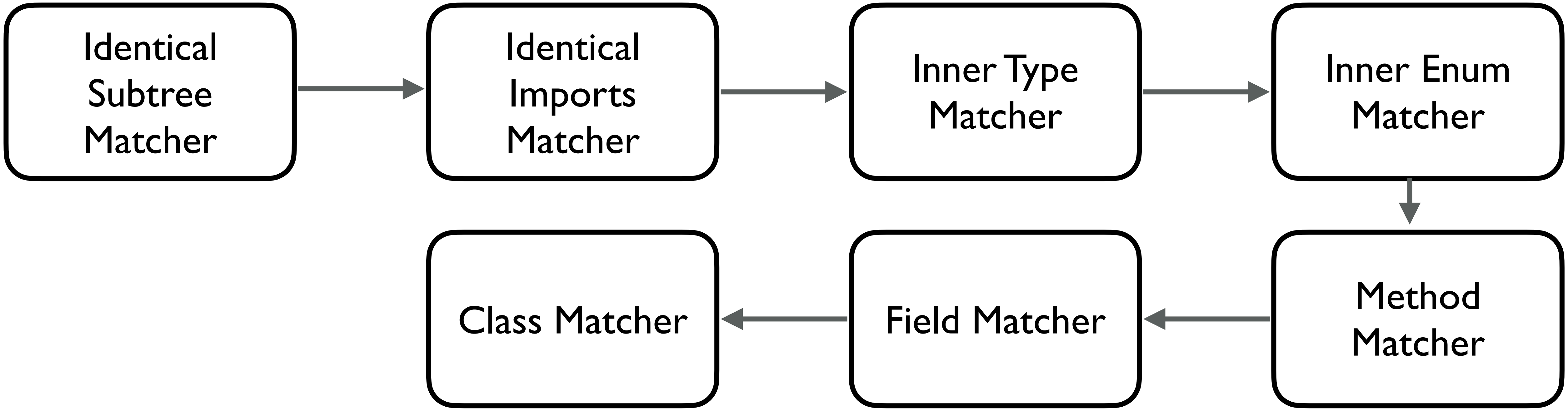
Improvements over GumTree

- Partial matching
- Merged name nodes
- Name-aware matching

Partial matching

Series of specialized matchers

Restricted scope per matcher



Merged name nodes

Merges name nodes with their parents

Reduces AST size

Prevents name mismatches

```
public class Test {
    public void foo() {
    }
}
```

```
public class Test {
    public void bar() {
        int foo = 1;
    }
}
```

Name-aware matching

Adding name-awareness to bottom-up phase of GumTree

Similarity of node names is taken into account

```
public class Test {
    public void foo() {
        if (!fields.isEmpty()) {
          //impl
        }
    }
}
```

```
public class Test {
    public void foo() {
        if (iterator.hasNext()) {
          //impl
        }
    }
}
```

Evaluation

Comparison between IJM, GumTree, and MtDiff

- Edit Script Size, Runtime, Accuracy, Helpfulness

10 open source Java Projects

- 61,040 commits, 392,492 revisions

307,081 revisions excluding JavaDoc and out of Memory revisions

Evaluation: edit script size

Evaluated all 307,081 revisions

	GumTree	MtDiff	IJM
Median Size	12	12	9

IJM has smallest edit script (alone or shared) in 95.22% of the revisions

GumTree in 53.08%
MtDiff in 54.53%

IJM ran on the reduced AST (merged name nodes) while MtDiff GumTree ran on the full AST

Effect statistically valid but negligible

Evaluation: accuracy

2400 randomly selected single edit actions evaluated

- 200 per action type and matcher

Classified as accurate/inaccurate

Criteria for accurate edit actions:

- Comprehensive
- Helpful
- No simpler solution

Evaluation: accuracy

MR: Misclassification Rate

NotA: Number of total actions

	GumTree		MtDiff		IJM	
	MR	NotA	MR	NotA	MR	NotA
Move	58.2%	720,303	81.5%	3,121,607	43.5%	510,250
Update	40%	938,288	37%	759,177	17%	503,423
Insert	5.5%	12,225,111	6%	9,642,897	5.5%	10,236,135
Delete	12%	5,478,973	11%	4,038,471	11.5%	5,021,193
Relative	10.98%		21.91%		8.9%	

Evaluation: helpfulness

11 independent external experts

3 randomly selected revisions per project

- Each revision consisting of ≥20 and ≤100 edit actions
- Including >= 1 move or update action

Each participant ranks the output of GumTree, IJM, and MtDiff according to helpfulness

Each participant evaluates one revision per project

Evaluation: helpfulness

	1st	2nd	3rd
GumTree	30	39	41
MtDiff	31	39	40
IJM	**49**	32	29

IJM ranks first in

49 out of 110 cases (44.5%)

18 out of 30 revisions (60%)

Pearson’s Chi^2 shows dependency between matcher and rankings

Hands on IJM and DiffViz

Veit Frick

Summary of results

IJM improves accuracy & helpfulness at no additional costs in runtime and edit script size

IJM on Github:

https://github.com/VeitFrick/IJM

DiffVizualizer: tool for navigating and visualizing diffs

https://github.com/W3D3/DiffVisualizer

Some references

Umldiff: An algorithm for object-oriented de- sign differencing. Xing et al. 2005

Change distilling: Tree differencing for fine-grained source code change extraction. Fluri et al. 2007

Fine-grained and Accurate Source Code Differencing. Falleri et al. 2014

Move-optimized source code tree differencing. Dotzler et al. 2016

Generating simpler AST edit scripts by considering copy-and-paste. Higo et al. 2017

Generating Accurate and Compact Edit Scripts Using Tree Differencing. Frick et al. 2018

DiffViz: A Diff Algorithm Independent Visualization Tool for Edit Scripts. V. Frick et al. 2018

ClDiff: generating concise linked code differences, Huang et al. 2018

FEVER: An Approach to Analyze Feature-Oriented Changes and Artefact Co-Evolution in Highly Configurable Systems, Dintzner et al. 2018