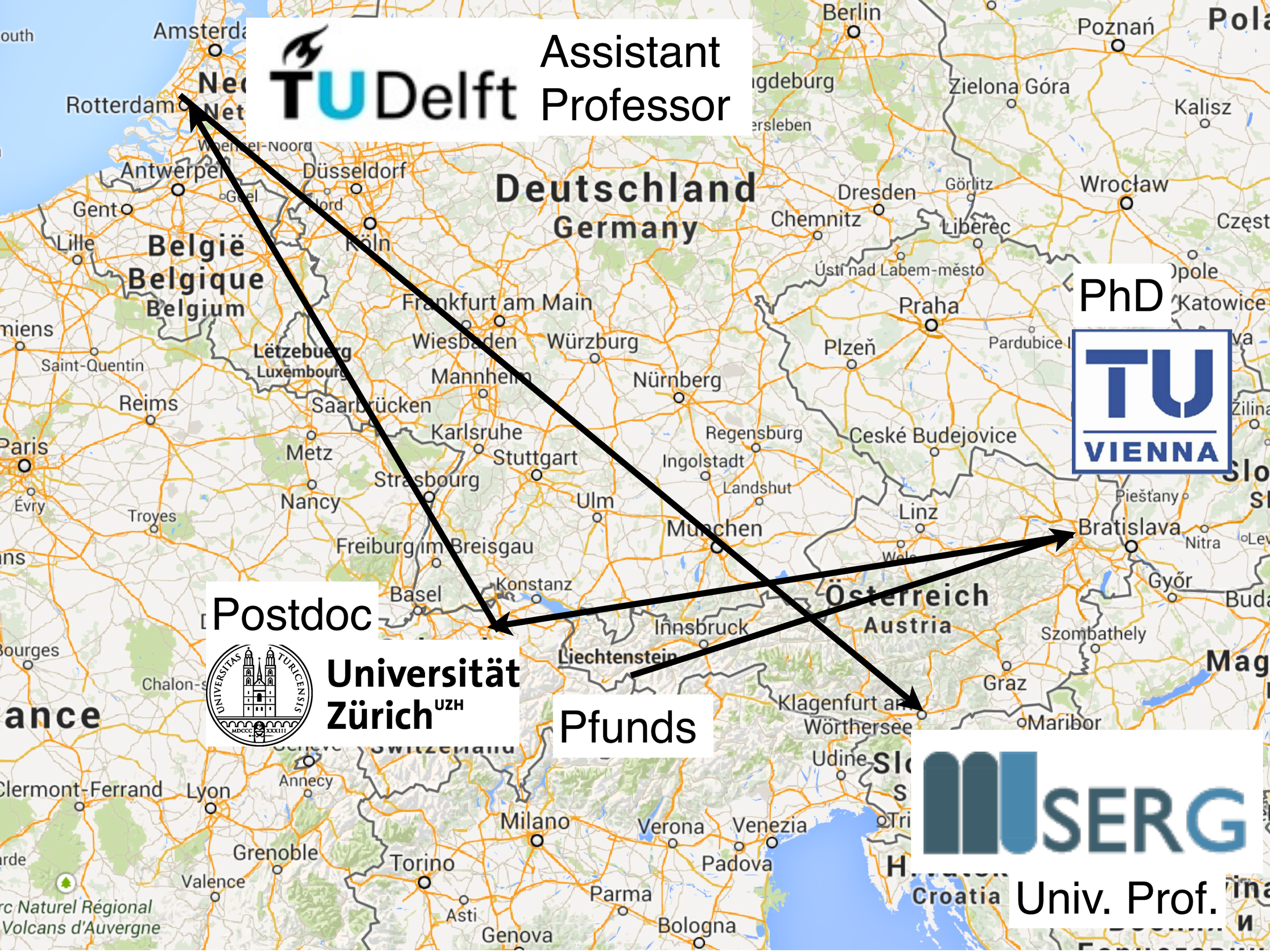
Mining and Analyzing Source Code Changes

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


TU Delft
Assistant Professor
PhD
TU VIENNA
Postdoc
Universität Zürich UZH
Pfunds
SERG
Univ. Prof.
Amsterdam
Rotterdam
Antwerpen
Gent
Lille
België
Belgique
Belgium
Düsseldorf
Köln
Deutschland
Germany
Berlin
Poznań
Zielona Góra
Kalisz
Dresden
Görlitz
Chemnitz
Liberec
Wrocław
Ústí nad Labem-město
Praha
Plzeň
Pardubice
Opole
Katowice
Frankfurt am Main
Wiesbaden
Würzburg
Mannheim
Nürnberg
Lëtzebuerg
Luxembourg
Saarbrücken
Karlsruhe
Stuttgart
Regensburg
Ingolstadt
Landshut
České Budějovice
Metz
Strasbourg
Nancy
Ulm
München
Linz
Bratislava
Nitra
Győr
Freiburg im Breisgau
Basel
Konstanz
Österreich
Austria
Innsbruck
Liechtenstein
Szombathely
Graz
Maribor
Klagenfurt am Wörthersee
Udine
Milano
Verona
Venezia
Padova
Torino
Parma
Bologna
Genova
Asti
Grenoble
Valence
Lyon
Annecy
Clermont-Ferrand
Saint-Quentin
Reims
Paris
Évry
Troyes
Bourges
Chalon-s
Croatia
Pieštany

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

Course overview

1. Fine-grained source code change extraction

   ChangeDistiller and IJM

   Hands on IJM and DiffViz

2. Using the fine-grained source code changes …

   Hands on analyzing the evolution of a system

   For bug prediction and change summarization

Lehman's Law of Software Evolution

Continuing change

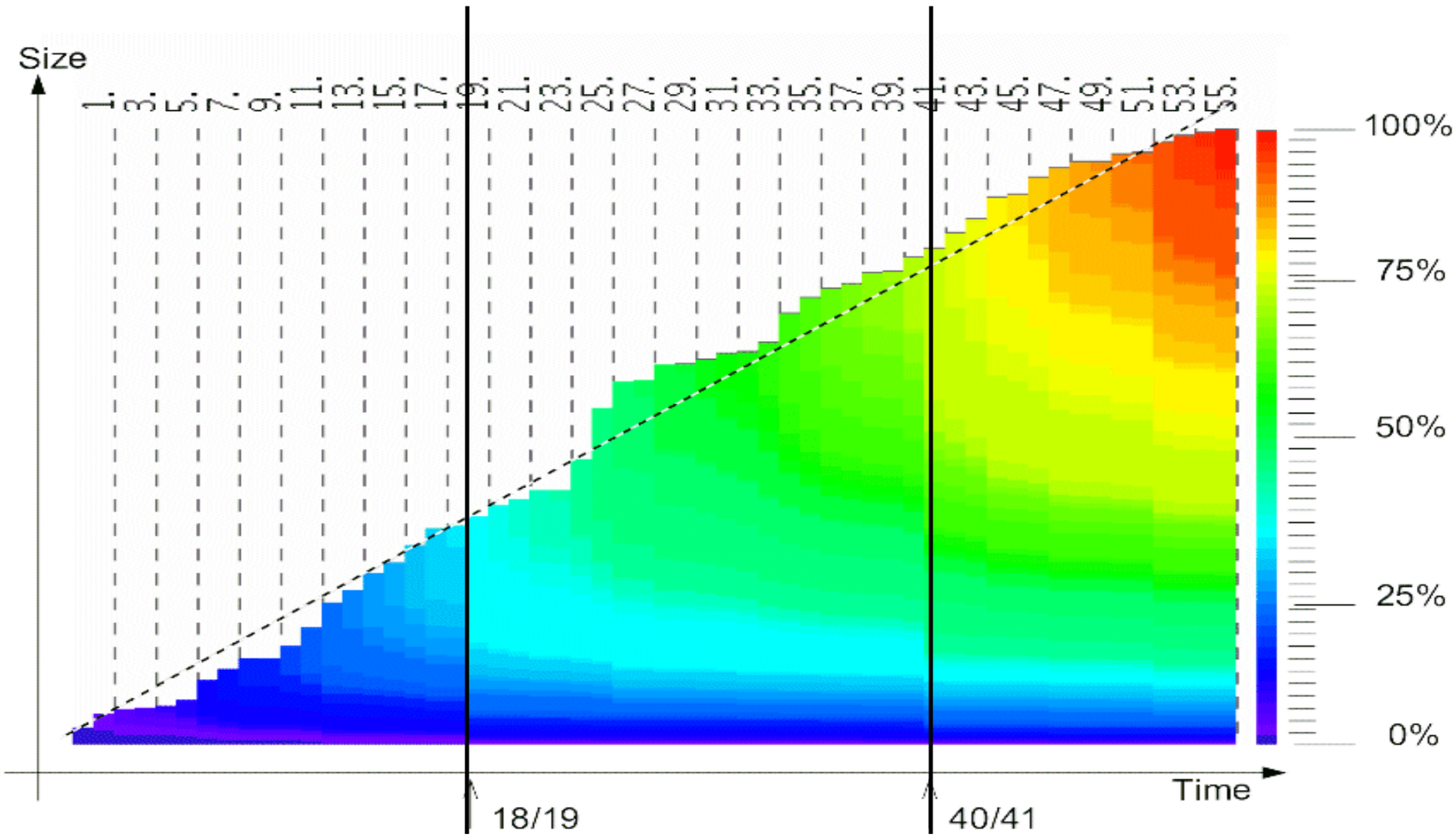
A program that is used in a real-world environment must change, or become progressively less useful in that environment.

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`

Previous Next

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

```
@@ -1029,10 +1029,15 @@ public void addDefaultListLayout(String moduleName, ViewContext<ThymeleafMetadat
1029  1029     @Override
1030  1030     protected boolean isUserManagedDocument(Document document) {
1031  1031
1032        -    Element root = document.getElementsByTag("html").get(0);
      1032  +    Elements match = document.getElementsByTag("html");
1033  1033
1034        -    if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
1035        -      return true;
      1034  +    if (match != null && match.size() > 0) {
      1035  +      Element root = match.get(0);
      1036  +      if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
      1037  +        return true;
      1038  +      }
      1039  +    } else {
      1040  +      return false;
1036  1041       }
1037  1042       return false;
1038  1043     }
```

0 comments on commit `095c25d`

What is the change impact?

File 1 of 1 in `095c25d`

Previous Next

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

```
@@ -1029,10 +1029,15 @@ public void addDefaultListLayout(String moduleName, ViewContext<ThymeleafMetadat
1029  1029      @Override
1030  1030      protected boolean isUserManagedDocument(Document document) {
1031  1031
1032        -     Element root = document.getElementsByTag("html").get(0);
      1032  +     Elements match = document.getElementsByTag("html");
1033  1033
1034        -     if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
1035        -       return true;
      1034  +     if (match != null && match.size() > 0) {
      1035  +       Element root = match.get(0);
      1036  +       if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
      1037  +         return true;
      1038  +       }
      1039  +     } else {
      1040  +       return false;
1036  1041        }
1037  1042        return false;
1038  1043      }
```

0 comments on commit `095c25d`

Do the changes affect my code?

File 1 of 1 in `095c25d`

Previous Next

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

```
@@ -1029,10 +1029,15 @@ public void addDefaultListLayout(String moduleName, ViewContext<ThymeleafMetadat
1029  1029      @Override
1030  1030      protected boolean isUserManagedDocument(Document document) {
1031  1031
1032        -     Element root = document.getElementsByTag("html").get(0);
      1032  +     Elements match = document.getElementsByTag("html");
1033  1033
1034        -     if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
1035        -       return true;
      1034  +     if (match != null && match.size() > 0) {
      1035  +       Element root = match.get(0);
      1036  +       if (root != null && root.hasAttr("data-z") && root.attr("data-z").equals("user-managed")) {
      1037  +         return true;
      1038  +       }
      1039  +     } else {
      1040  +       return false;
1036  1041        }
1037  1042        return false;
1038  1043      }
```

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

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

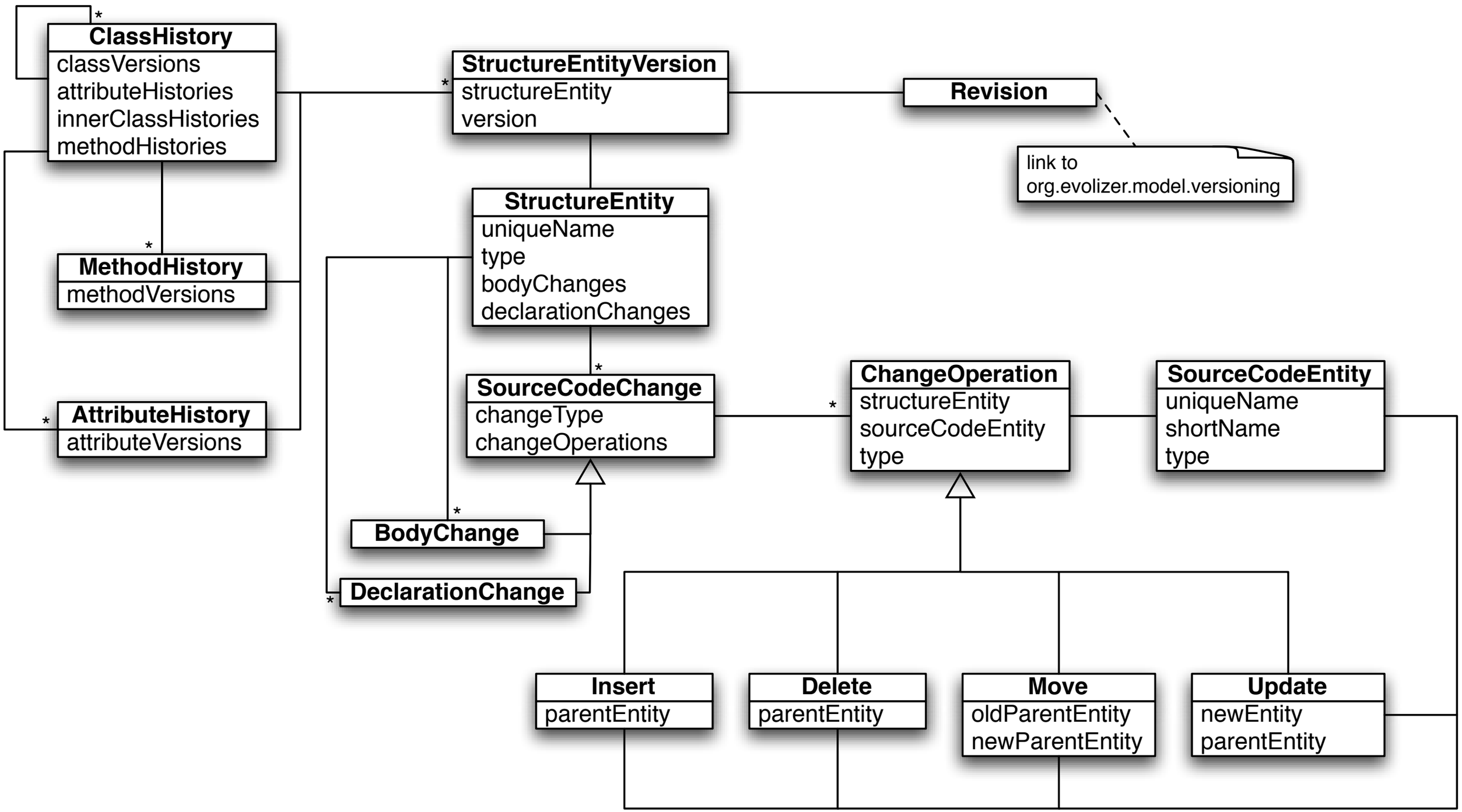
Using tree differencing, we can determine

- 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

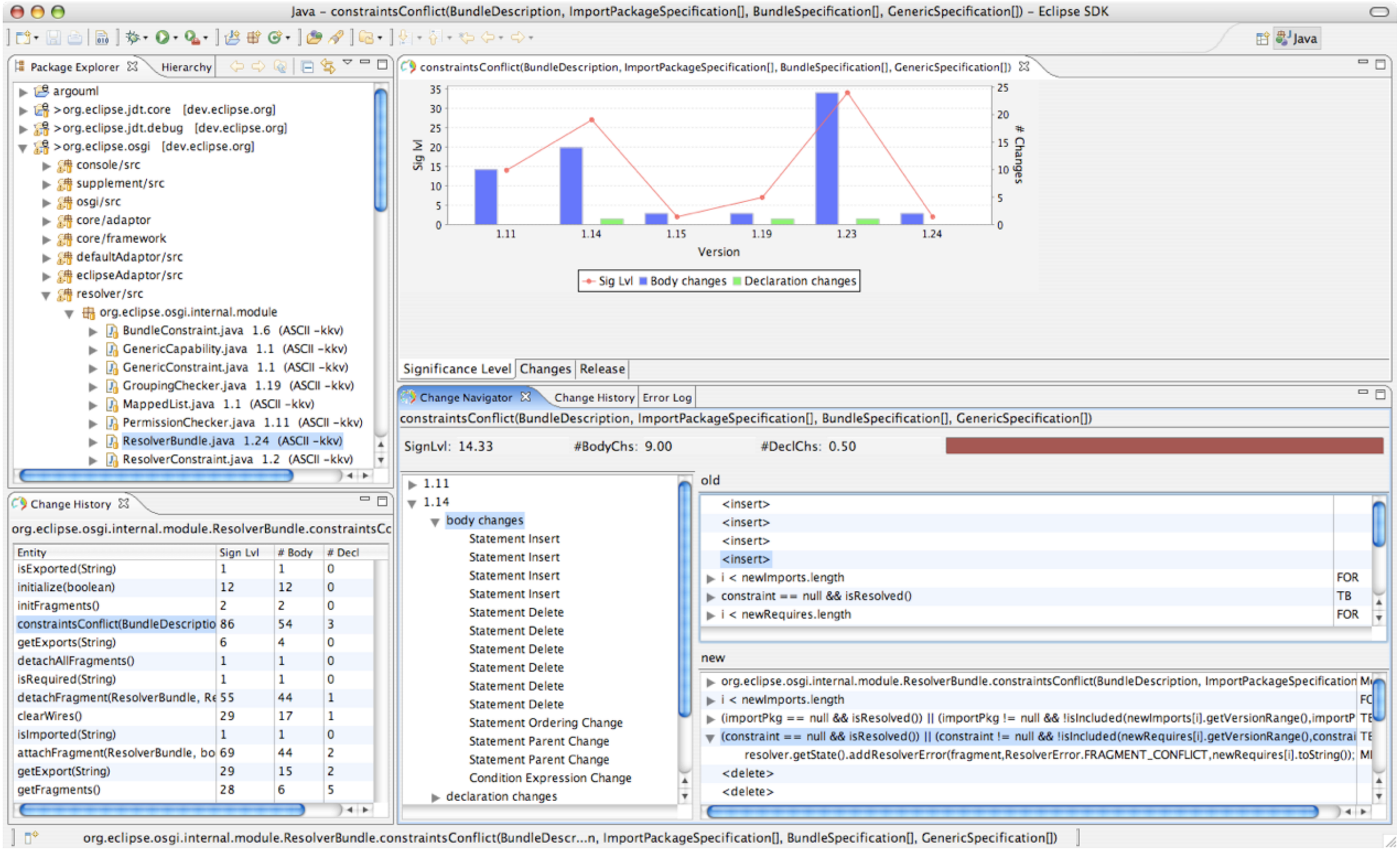
mDecl = changes to method declarations

stmt = insertion, deletion, ordering of executable statements

cond = changes to conditional expressions

else = insertion and deletion of else-parts

ChangeDistiller tool



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

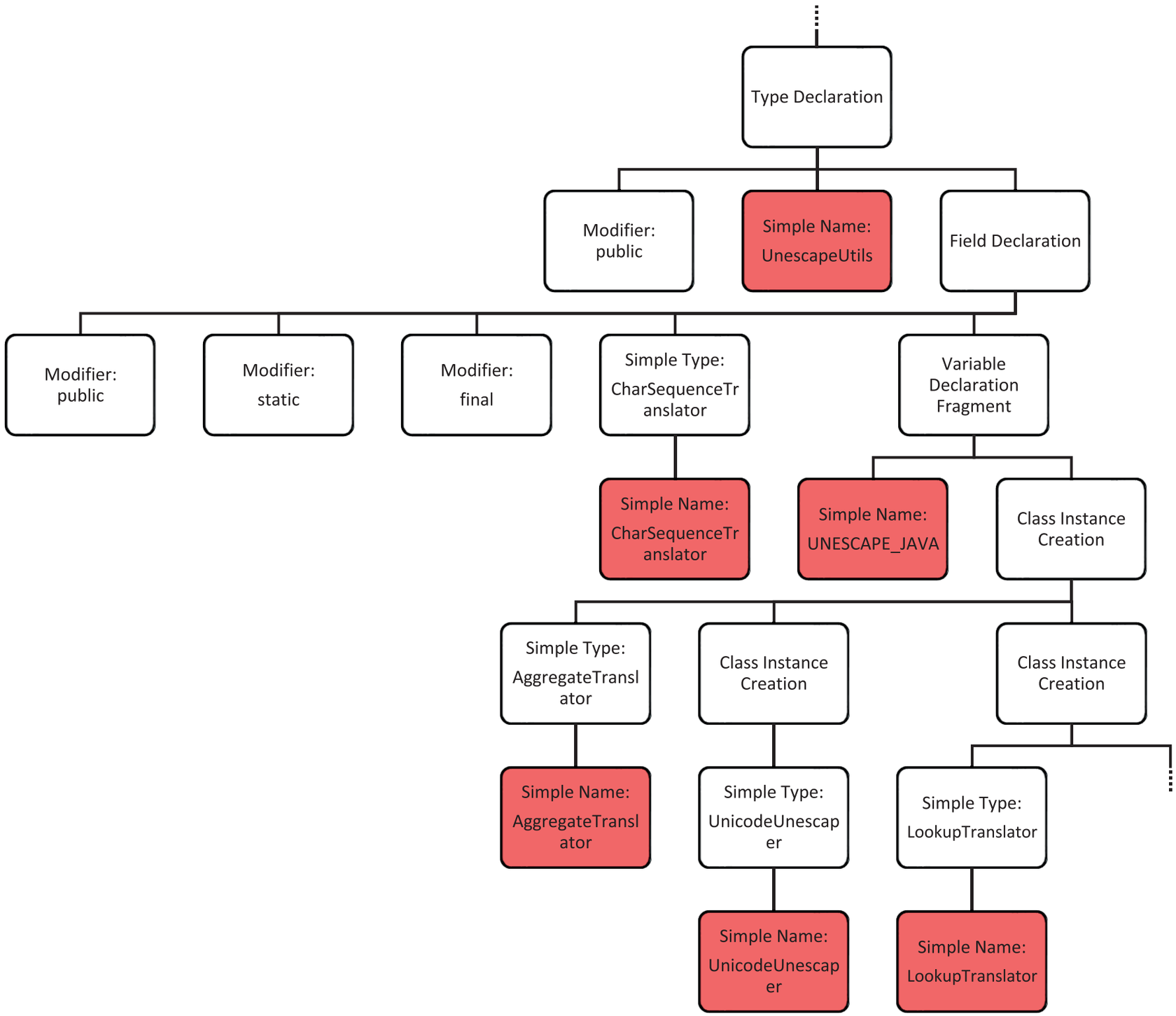
GumTree

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



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

Inner Type Matcher, Field Matcher, …

Merged name nodes

Merges name nodes with their parents

Reduces AST size

Prevents name mismatches

```
1. public class Test {
2.     public void foo() {
3.     }
4. }
```

```
1. public class Test {
2.     public void bar() {
3.         int foo = 1;
4.     }
5. }
```

Name-aware matching

Adding name-awareness to bottom-up phase of GumTree

Similarity of node names is taken into account

- Similarity threshold is a Levenshtein distance of < 0.3

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

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

DiffViz: tool for navigating and visualizing diffs

https://www.youtube.com/watch?v=RF93ey9GYoc

Research opportunities

Further improve the performance (precision) of the extraction algorithm(s)

Extract changes of dependencies (our current work)

- E..g, consider changes in call, access, inheritance, and type dependencies

Integrate and visualize changes

- Extract changes from other source files, e.g., configuration files, project and build files (FEVER)
- Integrate and visualize them to allow engineers to better understand them

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

CIDiff: 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

Hands on IJM and DiffViz

Hands on analyzing the evolution of a system

Sources at: https://github.com/pinzger/siesta2019