powered by Cha-Q

Cha-Q Search-and-replace

Research on Repeating changes

Problem: performing systematic edits within and across programs

before

```
public class BreakStatement extends Statement {
    @EntityProperty(value = SimpleName.class)
    private EntityIdentifier label;

    public EntityIdentifier getLabel() {
        return label;
    }

    public void setLabel(EntityIdentifier label) {
        this.label = label;
    }
}
```

value of annotation

after

```
public class BreakStatement extends Statement {
    @EntityProperty(value = SimpleName.class)
    private EntityIdentifier<SimpleName> label;

    public EntityIdentifier<SimpleName> getLabel() {
        return label;
    }

    public void setLabel(EntityIdentifier<SimpleName> label) {
        this.label = label;
    }
}
```

to be used as type parameter

find and fix all occurrences of the same bug

evolve existing code to newer Java versions

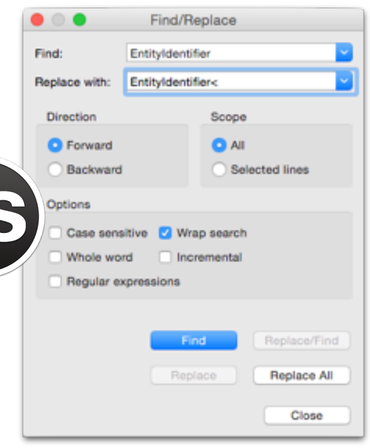
update existing clients to new version of API

co-maintain multiple system variants

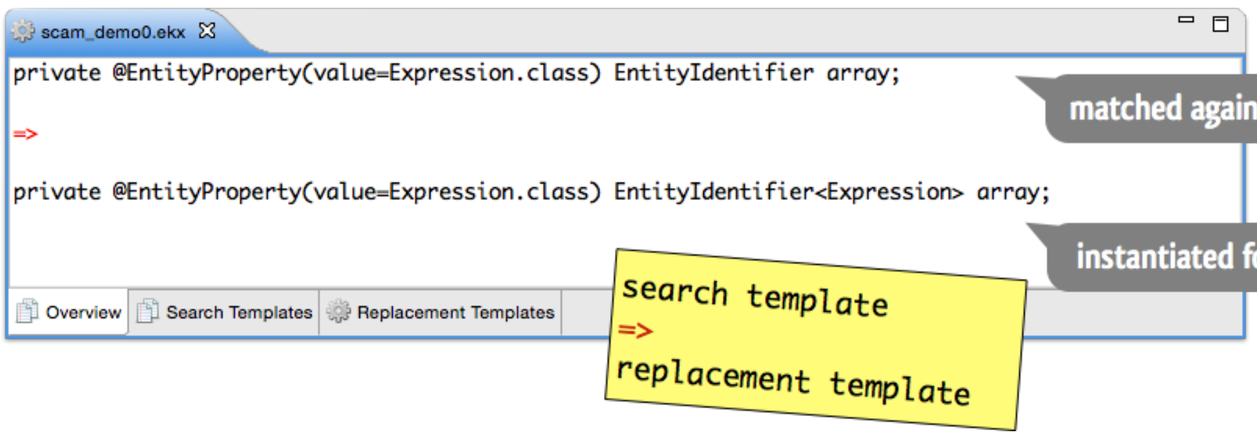
```
;;relies on functions for program manipulation
;;can be invoked directly
;;following fixes all at once
(for [[field fieldtype annotype]
      ;;subjects of change
      (ekeko [?field ?fieldtype ?annotype]
             (fresh [?anno ?annotypeliteral]
                    (field-declaration!incorrect ?field)
                    (has :type ?field ?fieldtype)
                    (fielddeclaration-annotation ?field ?anno)
                    (annotation!ep-typeliteral ?anno ?annotypeliteral)
                    (has :type ?annotypeliteral ?annotype)))]
  ;;actual changes
  (let [fieldtype-copy (copy-astnode fieldtype)
        annotype-copy (copy-astnode annotype)
        new-type (create-parameterized-type fieldtype-copy)]
    (change-property-node field :type new-type)
    (add-node (current-rewrite-for-cu (.getRoot field)) new-type :typeArguments annotype-copy 0)))

;;have scheduled changes
;;still need to apply them
(apply-and-reset-rewrites)
```

VS



Approach: search-and-replace using code templates



matched against code

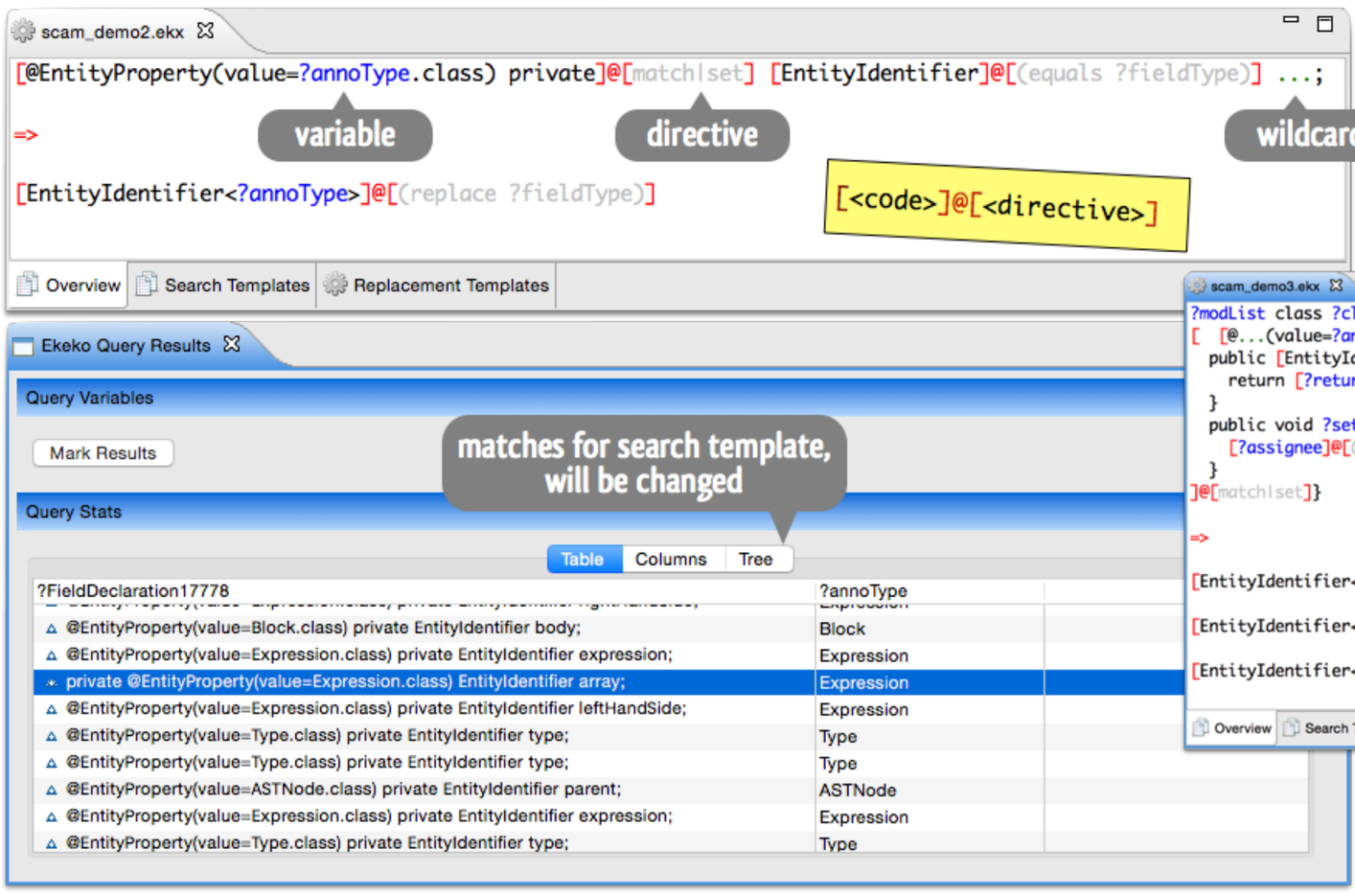
instantiated for each match

BUT:

search template
has only one match

replacement template
replaces entire match

Innovation: directives for control over matching and replacing



matches control and data flow variants!

```
scam_demo3.ekx
?modlist class ?className {
[  [@...(value=?annoType.class) private]@[match|set] [EntityIdentifier]@[child* (equals ?fieldType)] ?field;
  public [EntityIdentifier]@[child* (equals ?returnType)] ?getterName(){
    return [?returned]@[(refers-to ?field)];
  }
  public void ?setterName(  [EntityIdentifier]@[child* (equals ?paramType)] ?param){
    [?assignee]@[(refers-to ?field)]=[?assigned]@[(refers-to ?param)];
  }
]@[match|set]}

=>

[EntityIdentifier<?annoType>]@[(replace ?fieldType)]

[EntityIdentifier<?annoType>]@[(replace ?paramType)]

[EntityIdentifier<?annoType>]@[(replace ?returnType)]
```

Overview | Search Templates | Replacement Templates

PDF

The Ekeko/X Program Transformation Tool
Coen De Roover and Katsuro Inoue
Proceedings of 14th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM2014)

http://soft.vub.ac.be/chaq

Vrije Universiteit Brussel

Universiteit Antwerpen

iMT