CIDE+: A Semi-automatic Approach for Extracting Software Product Lines

Virgilio Borges, Rógel Garcia,
Marco Túlio Valente

DCC – UFMG

CBSOFT 2010 - Tools Session

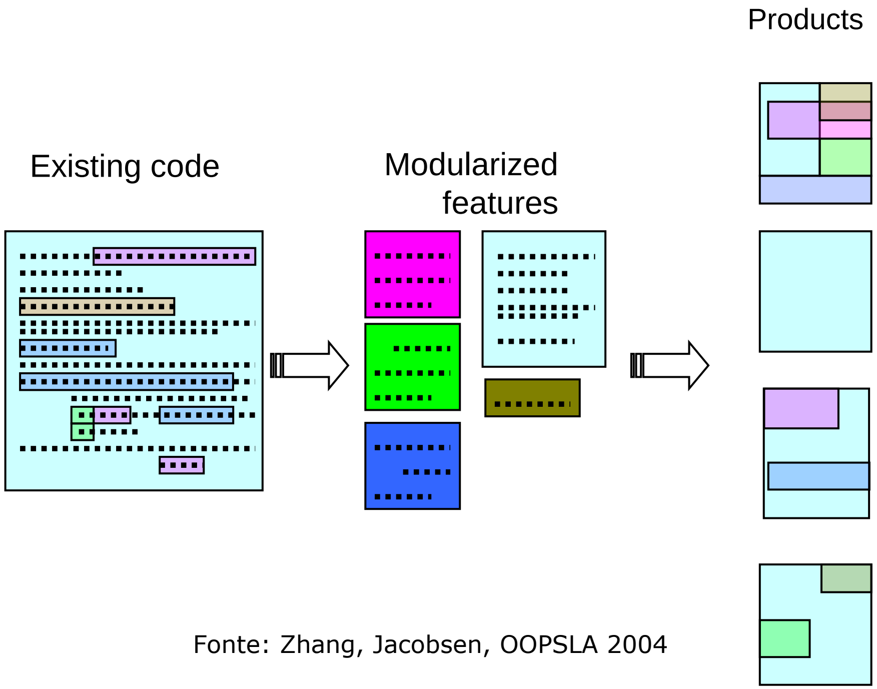
Motivation

- SPL extraction is a time-consuming task
  - Even starting from an existing codebase

- Two approaches for extracting SPL:
  - Compositional-based
  - Annotation-based

Compositional-based Approaches

- Example: aspects (AspectJ)



Fonte: Zhang, Jacobsen, OOPSLA 2004

- Physical (“real”) modularization and separation of concerns
- Problem: costs do not outweight the benefits

Annotation-based Approaches

- Example: preprocessors

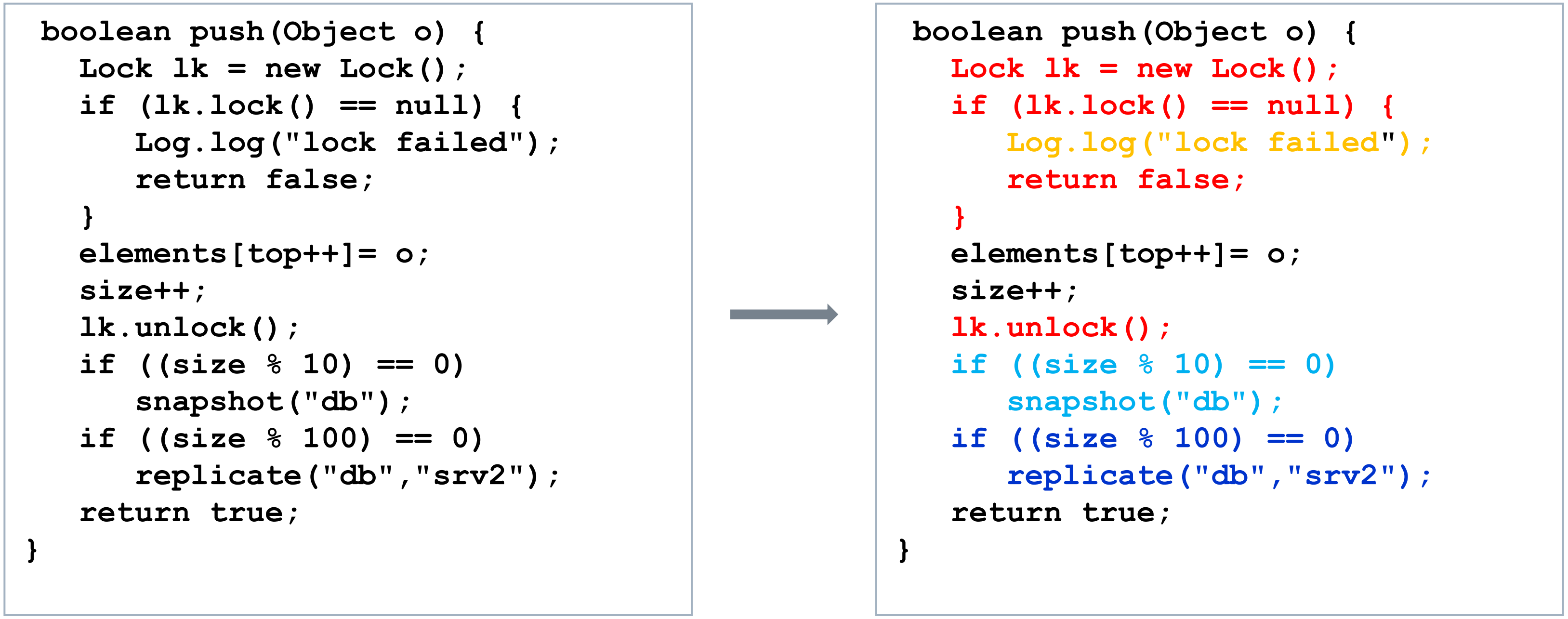
```
 boolean push(Object o) {
   Lock lk = new Lock();
   if (lk.lock() == null) {
      Log.log("lock failed");
      return false;
   }
   elements[top++]= o;
   size++;
   lk.unlock();
   if ((size % 10) == 0)
      snapshot("db");
   if ((size % 100) == 0)
      replicate("db","srv2");
   return true;
}
```

```
boolean push(Object o) {
   #ifdef MULTITHREADING
   Lock lk = new Lock();
   if (lk.lock() == null) {
      #ifdef LOGGING
      Log.log("lock failed");
      #endif
      return false;
   }
   #endif
   elements[top++]= o;
   size++;
   #ifdef MULTITHREADING
   lk.unlock();
   #endif
   #ifdef SNAPSHOT
   if ((size % 10) == 0)
      snapshot("db");
   #endif
   #ifdef REPLICATION
   if ((size % 100) == 0)
      replicate("db","srv2");
   #endif
   return true;
}
```

- It works. It is widely used.
- Problems: annotation hell; code pollution

Visual Annotations

- CIDE: Colored IDE (Eclipse + background colors)



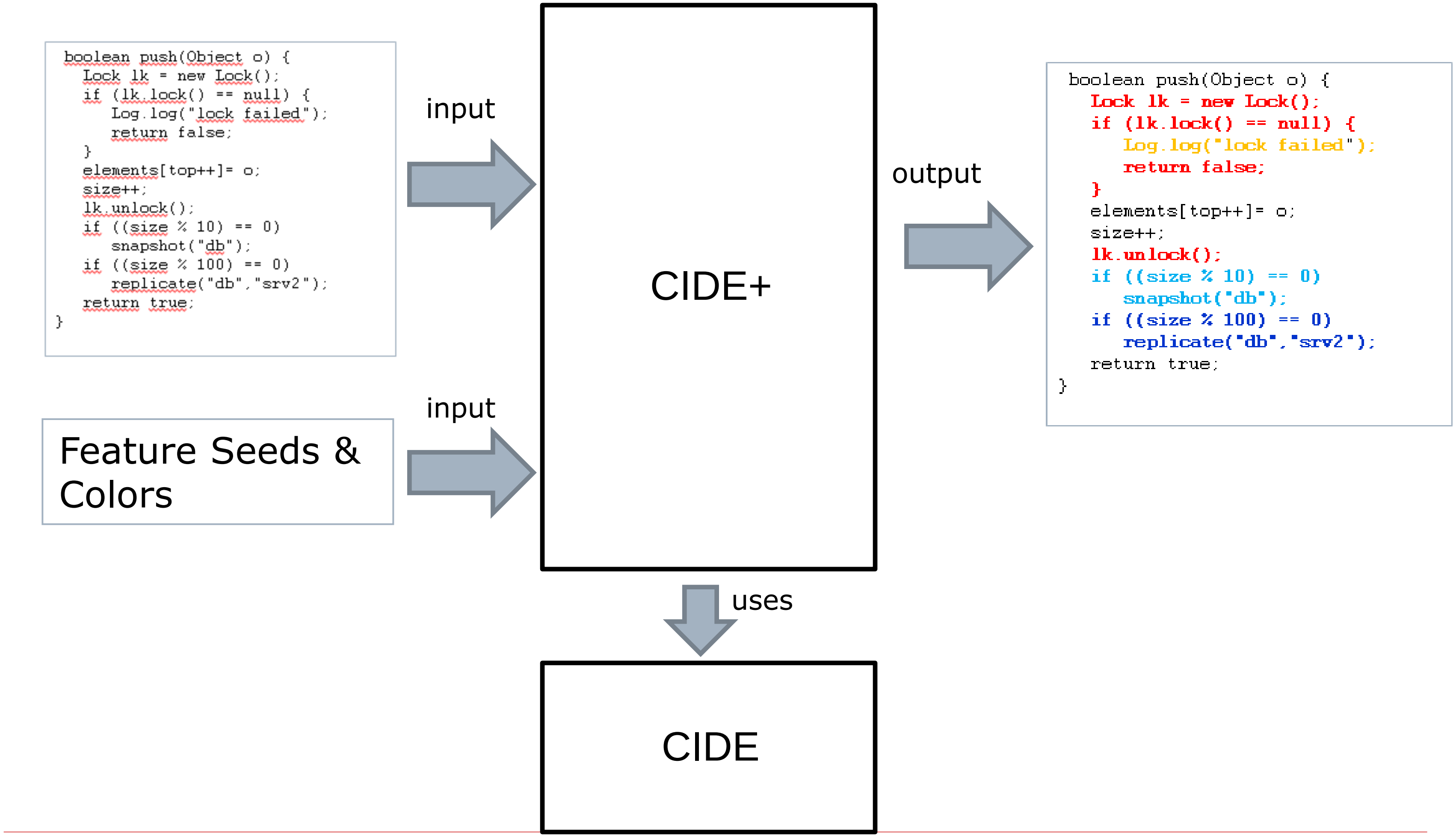
```
 boolean push(Object o) {
   Lock lk = new Lock();
   if (lk.lock() == null) {
      Log.log("lock failed");
      return false;
   }
   elements[top++]= o;
   size++;
   lk.unlock();
   if ((size % 10) == 0)
      snapshot("db");
   if ((size % 100) == 0)
      replicate("db","srv2");
   return true;
}
```

```
 boolean push(Object o) {
   Lock lk = new Lock();
   if (lk.lock() == null) {
      Log.log("lock failed");
      return false;
   }
   elements[top++]= o;
   size++;
   lk.unlock();
   if ((size % 10) == 0)
      snapshot("db");
   if ((size % 100) == 0)
      replicate("db","srv2");
   return true;
}
```

- It works, generating less code poluttion than #ifdefs
- Problem: colors assigned manually (repetitive, error-prone etc)

Our Tool: CIDE+

- Semi-automatic approach to assign colors to feature code



```
boolean push(Object o) {
   Lock lk = new Lock();
   if (lk.lock() == null) {
      Log.log("lock failed");
      return false;
   }
   elements[top++]= o;
   size++;
   lk.unlock();
   if ((size % 10) == 0)
      snapshot("db");
   if ((size % 100) == 0)
      replicate("db","srv2");
   return true;
}
```

input

Feature Seeds & Colors

input

CIDE+

output

```
boolean push(Object o) {
   Lock lk = new Lock();
   if (lk.lock() == null) {
      Log.log("lock failed");
      return false;
   }
   elements[top++]= o;
   size++;
   lk.unlock();
   if ((size % 10) == 0)
      snapshot("db");
   if ((size % 100) == 0)
      replicate("db","srv2");
   return true;
}
```

uses

CIDE

Input: Feature Seeds

- Program elements that implement an optional feature F
- When F is disabled, S can be removed from the code
- Example: logging

seed

```
void log(String s) {
   .....
}
```

- Granularity: package, class, method, field
- Therefore, cannot dispense a meaningful knowledge about the internals of the target system

Annotation Algorithm

- Fixed point algorithm
- Two phases:
  - Color Propagation
  - Color Expansion

1st Phase: Propagation

- Marks all program elements that reference the seeds S

seed

```
void log(String s) {
   .....
}
```

references to the seeds

```
{
  log("stack overflow");
  .....
  .....
  log("lock failed");
  .....
  log("page delivered");
  .....
  .....
  .....

  log("new customer inserted");
  .....
  log("game over");
  .....
  .....
  log("user authenticated");
  .....
}
```

1st Phase: Propagation

- Marks all program elements that reference the seeds S

seed

```
void log(String s) {
   .....
}
```

references to the seeds

```
{
  log("stack overflow");
  .....
  .....
  log("lock failed");
  .....
  log("page delivered");
  .....
  .....
  .....

  log("new customer inserted");
  .....
  log("game over");
  .....
  .....
  log("user authenticated");
  .....
}
```

Color Propagation Rules

```
$ColorPropagation(Package\ p, Color\ c) =$
    $\forall t \in classes(p) \rightarrow ColorPropagation(t, c);$
    $\forall i \in interfaces(p) \rightarrow ColorPropagation(i, c);$
    $\forall p = \texttt{import(p.}*\texttt{)} \rightarrow cide(p, c);$
```

Pacotes

```
$ColorPropagation(Class\ t, Color\ c) =$
    $\forall m \in meths(t) \rightarrow ColorPropagation(m, c);$
    $\forall f \in fields(t) \rightarrow ColorPropagation(f, c);$
    $\forall s \in extends(t) \rightarrow ColorPropagation(s, c);$
    $\forall v \in hasType(t) \rightarrow ColorPropagation(v, c);$
    $\forall m \in hasReturnType(t) \rightarrow ColorPropagation(m, c);$
    $\forall n = \texttt{new(t)} \rightarrow cide(n, c);$
    $\forall p = \texttt{import(t)} \rightarrow cide(p, c);$
```

Classes

```
$ColorPropagation(Interface\ i, Color\ c) =$
    $p = \texttt{declaration(i)} \rightarrow cide(p, c);$
    $\forall t \in impl(i) \wedge \forall m \in meths(t) \wedge m \in i \rightarrow ColorPropagation(m, c);$
    $\forall t \in hasType(i) \rightarrow ColorPropagation(t, c);$
    $\forall m \in hasReturnType(t) \rightarrow ColorPropagation(m, c);$
    $\forall p = \texttt{import(i)} \rightarrow cide(p, c);$
```

Interfaces

Color Propagation Rules

```
$ColorPropagation(Method\ m, Color\ c) =$
    $p = \texttt{impl(m)} \rightarrow cide(p, c);$
    $\forall s = \texttt{call(m)} \rightarrow cide(s, c);$
    $\forall m' \in \texttt{overrides(m)} \rightarrow ColorPropagation(m', c).$
```

→ Métodos

```
$ColorPropagation(Field\ f, Color\ c) =$
    $d = \texttt{declaration(f)} \rightarrow cide(d, c);$
    $\forall s = \texttt{access(f)} \rightarrow cide(s, c).$
```

→ Campos

```
$ColorPropagation(LocalVariable\ i, Color\ c) =$
    $d = \texttt{declaration(i)} \rightarrow cide(d, c);$
    $\forall s = \texttt{access(i)} \rightarrow cide(s, c).$
```

→ Var. locais

```
$ColorPropagation(FormalParam\ p, Color\ c) =$
    $d = \texttt{declaration(p)} \rightarrow cide(d, c);$
```

→ Parâmetros

2nd Phase: Color Expansion

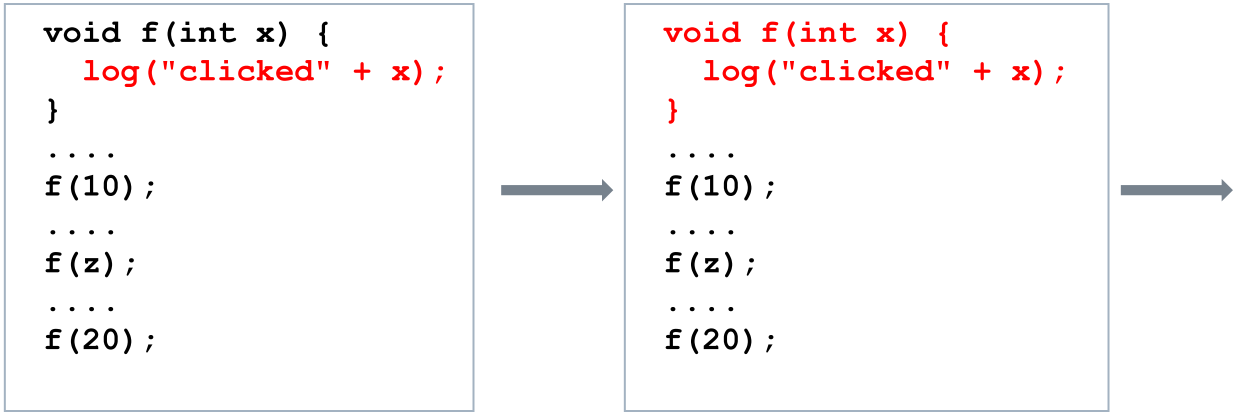
- Checks whether the enclosing context of the elements annotated in the previous phase can also be marked.

```
void f(int x) {
  log("clicked" + x);
}
....
f(10);
....
f(z);
....
f(20);
```

after color propagation

2nd Phase: Color Expansion

- Checks whether the enclosing context of the elements annotated in the previous phase can also be marked.



```
void f(int x) {
  log("clicked" + x);
}
....
f(10);
....
f(z);
....
f(20);
```

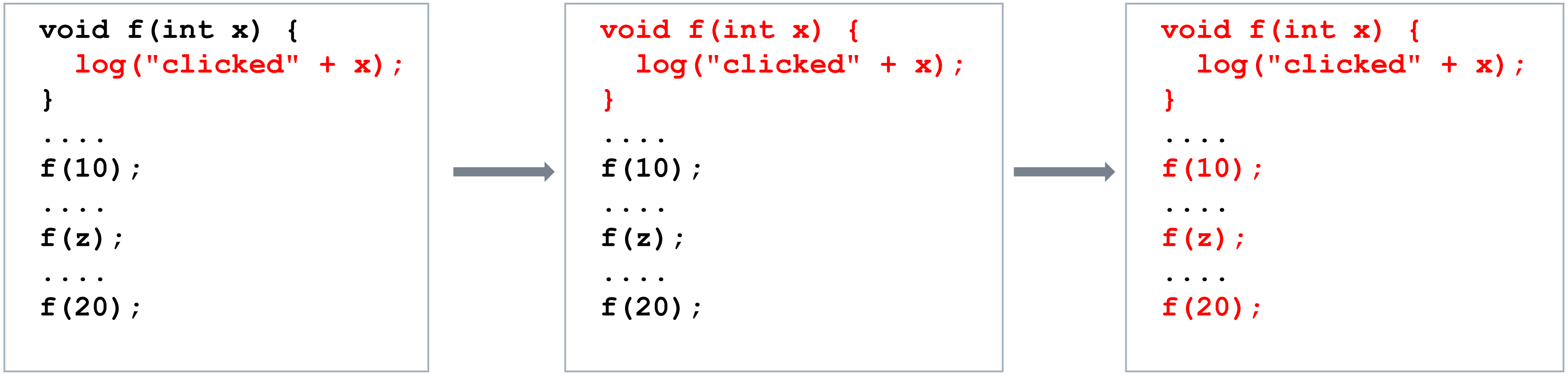
after color propagation

```
void f(int x) {
  log("clicked" + x);
}
....
f(10);
....
f(z);
....
f(20);
```

color expansion

2nd Phase: Color Expansion

- Checks whether the enclosing context of the elements annotated in the previous phase can also be marked.



```
void f(int x) {
  log("clicked" + x);
}
....
f(10);
....
f(z);
....
f(20);
```

after color propagation

```
void f(int x) {
  log("clicked" + x);
}
....
f(10);
....
f(z);
....
f(20);
```

color expansion

```
void f(int x) {
  log("clicked" + x);
}
....
f(10);
....
f(z);
....
f(20);
```

expansion can trigger propagation

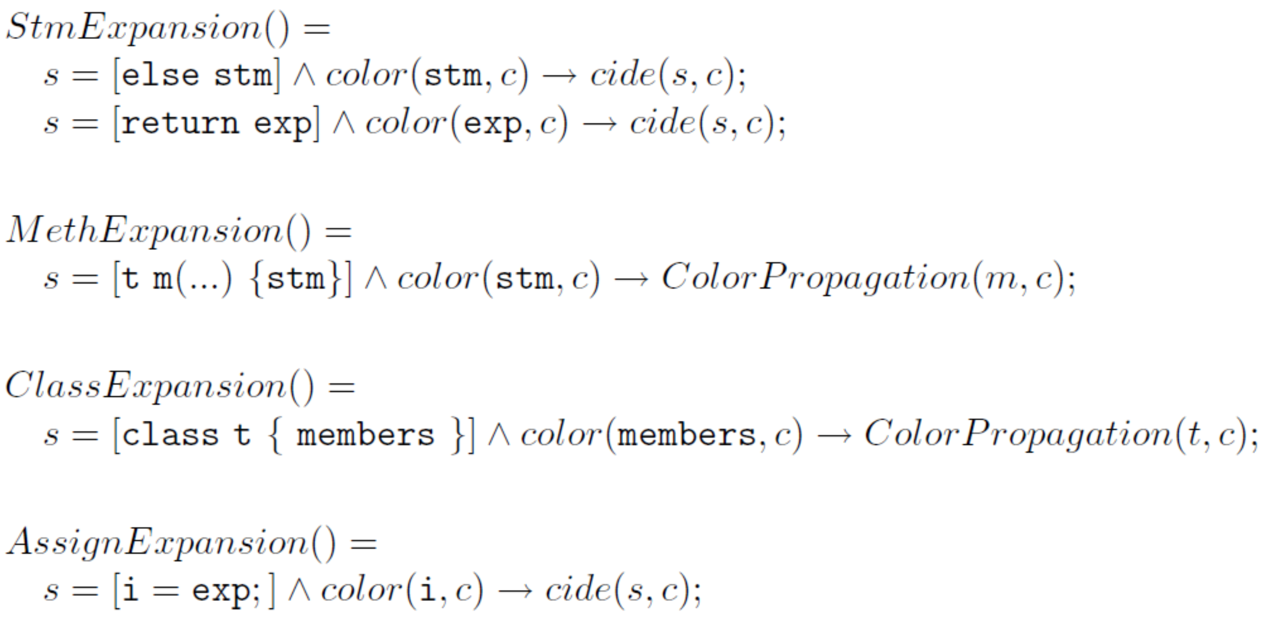
Color Expansion Rules

$$
\begin{array}{l}
\mathit{BodyExpansion}() = \\
\quad s = [\mathtt{if(exp)\ stm}] \wedge \mathit{color}(\mathtt{exp}, c) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{while\ exp\ stm}] \wedge \mathit{color}(\mathtt{exp}, c) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{do\ stm\ while\ exp}] \wedge \mathit{color}(\mathtt{exp}, c) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{case(exp)\ \{stm\}}] \wedge \mathit{color}(\mathtt{exp}, c) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{switch(exp)\ \{stm\}}] \wedge \mathit{color}(\mathtt{exp}, c) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{for(e_1; e_2; e_3)\ stm}] \wedge \mathit{color}(\mathtt{e_1}, c) \wedge \mathit{color}(\mathtt{e_2}, c) \wedge \mathit{color}(\mathtt{e_3}, c) \rightarrow \mathit{cide}(s, c);
\end{array}
$$

$$
\begin{array}{l}
\mathit{ExpExpansion}() = \\
\quad s = [\mathtt{if(exp)\ stm}] \wedge \mathit{color}(\mathtt{stm}, c) \wedge \mathit{free}(\mathtt{exp}) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{while\ exp\ stm}] \wedge \mathit{color}(\mathtt{stm}, c) \wedge \mathit{free}(\mathtt{exp}) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{do\ stm\ while\ exp}] \wedge \mathit{color}(\mathtt{stm}, c) \wedge \mathit{free}(\mathtt{exp}) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{if(exp)\ s_1\ else\ s_2}] \wedge \mathit{color}(\mathtt{s_1}, c) \wedge \mathit{color}(\mathtt{s_2}, c) \wedge \mathit{free}(\mathtt{exp}) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{case(exp)\ \{stm\}}] \wedge \mathit{color}(\mathtt{stm}, c) \wedge \mathit{free}(\mathtt{exp}) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{switch(exp)\ \{stm\}}] \wedge \mathit{color}(\mathtt{stm}, c) \wedge \mathit{free}(\mathtt{exp}) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{for(e_1; e_2; e_3)\ stm}] \wedge \mathit{color}(\mathtt{stm}, c) \wedge \mathit{free}(\mathtt{e_1}, \mathtt{e_2}, \mathtt{e_3}) \rightarrow \mathit{cide}(s, c);
\end{array}
$$

$$
\begin{array}{l}
\mathit{StmExpansion}() = \\
\quad s = [\mathtt{else\ stm}] \wedge \mathit{color}(\mathtt{stm}, c) \rightarrow \mathit{cide}(s, c); \\
\quad s = [\mathtt{return\ exp}] \wedge \mathit{color}(\mathtt{exp}, c) \rightarrow \mathit{cide}(s, c);
\end{array}
$$

Color Expansion Rules



$$
\begin{aligned}
&StmExpansion() = \\
&\quad s = [\texttt{else stm}] \wedge color(\texttt{stm}, c) \rightarrow cide(s, c); \\
&\quad s = [\texttt{return exp}] \wedge color(\texttt{exp}, c) \rightarrow cide(s, c);
\end{aligned}
$$

$$
\begin{aligned}
&MethExpansion() = \\
&\quad s = [\texttt{t m(...) \{stm\}}] \wedge color(\texttt{stm}, c) \rightarrow ColorPropagation(m, c);
\end{aligned}
$$

$$
\begin{aligned}
&ClassExpansion() = \\
&\quad s = [\texttt{class t \{ members \}}] \wedge color(\texttt{members}, c) \rightarrow ColorPropagation(t, c);
\end{aligned}
$$

$$
\begin{aligned}
&AssignExpansion() = \\
&\quad s = [\texttt{i} = \texttt{exp;}\,] \wedge color(\texttt{i}, c) \rightarrow cide(s, c);
\end{aligned}
$$

Semi-automatic Expansions

- In some situations, the algorithm can lead to type/syntatic errors

syntax error

```
if (option == k) {
    ....
}
```

type errors

```
 T foo() {
   ...    // no returns
   return t;
 }
```

- In other situations, it is complex to infer if expansion is safe

after propagation

```
if (bar()) {
    stm;
}
x= y;
```

→

Semi-automatic Expansions

- In some situations, the algorithm can lead to type/syntatic errors

syntax error

```
if (option == k) {
    ....
}
```

type errors

```
T foo() {
   ...     // no returns
   return t;
}
```

- In other situations, it is complex to infer if expansion is safe

after propagation

```
if (bar()) {
    stm;
}
x= y;
```

→

expansion

```
if (bar()) {
    stm;
}
x= y;
```

Semi-automatic Expansions

- In some situations, the algorithm can lead to type/syntatic errors

syntax error

```
if (option == k) {
    ....
}
```

type errors

```
 T foo() {
   ...     // no returns
   return t;
 }
```

- In other situations, it is complex to infer if expansion is safe

after propagation

```
if (bar()) {
    stm;
}
x= y; // bar() updates y
```

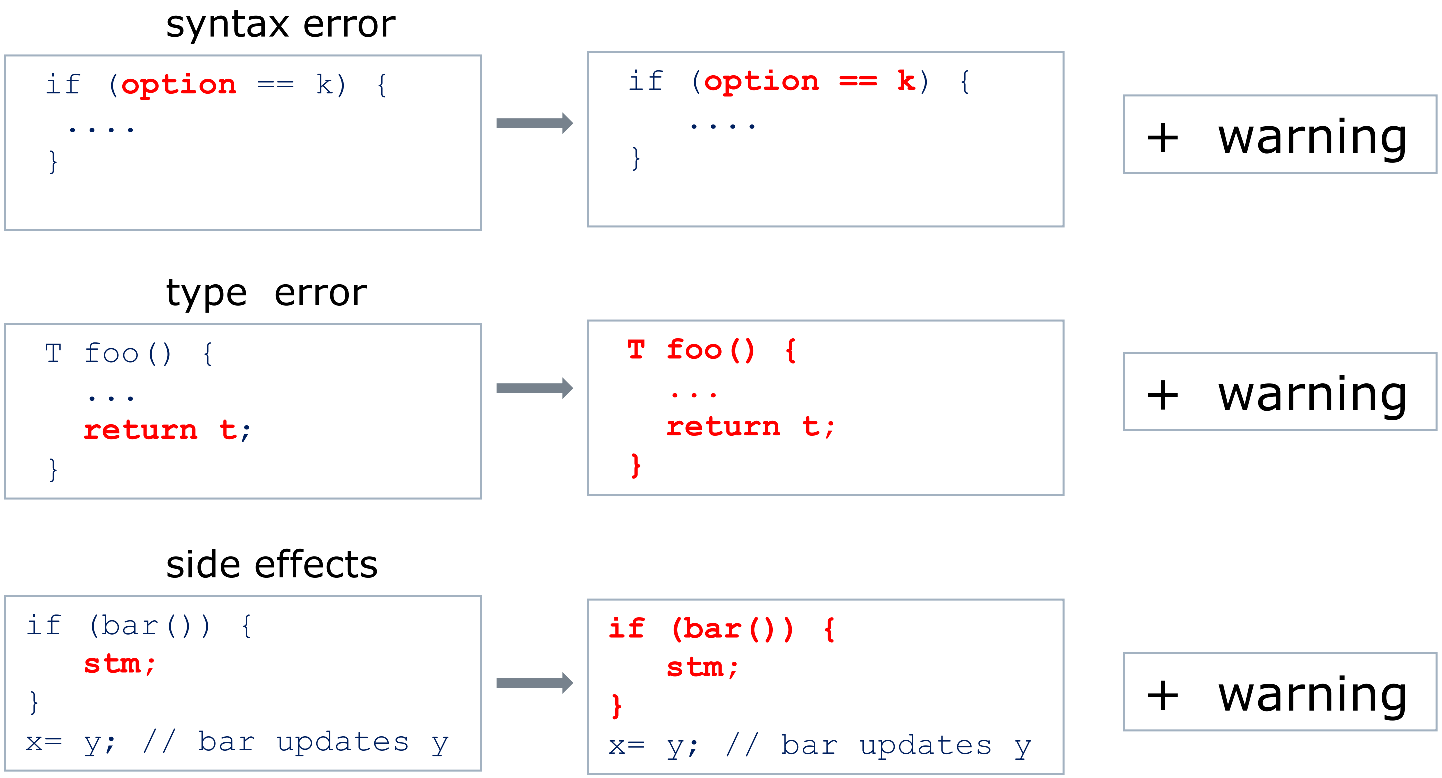
↛

unsafe expansion

```
if (bar()) {
    stm;
}
x= y; // bar() updates y
```

Semi-Automatic Expansions

- In the previous cases, we apply a default expansion (and generate a warning)



syntax error

```
if (option == k) {
 ....
}
```

```
if (option == k) {
   ....
}
```

+ warning

type error

```
T foo() {
  ...
  return t;
}
```

```
T foo() {
  ...
  return t;
}
```

+ warning

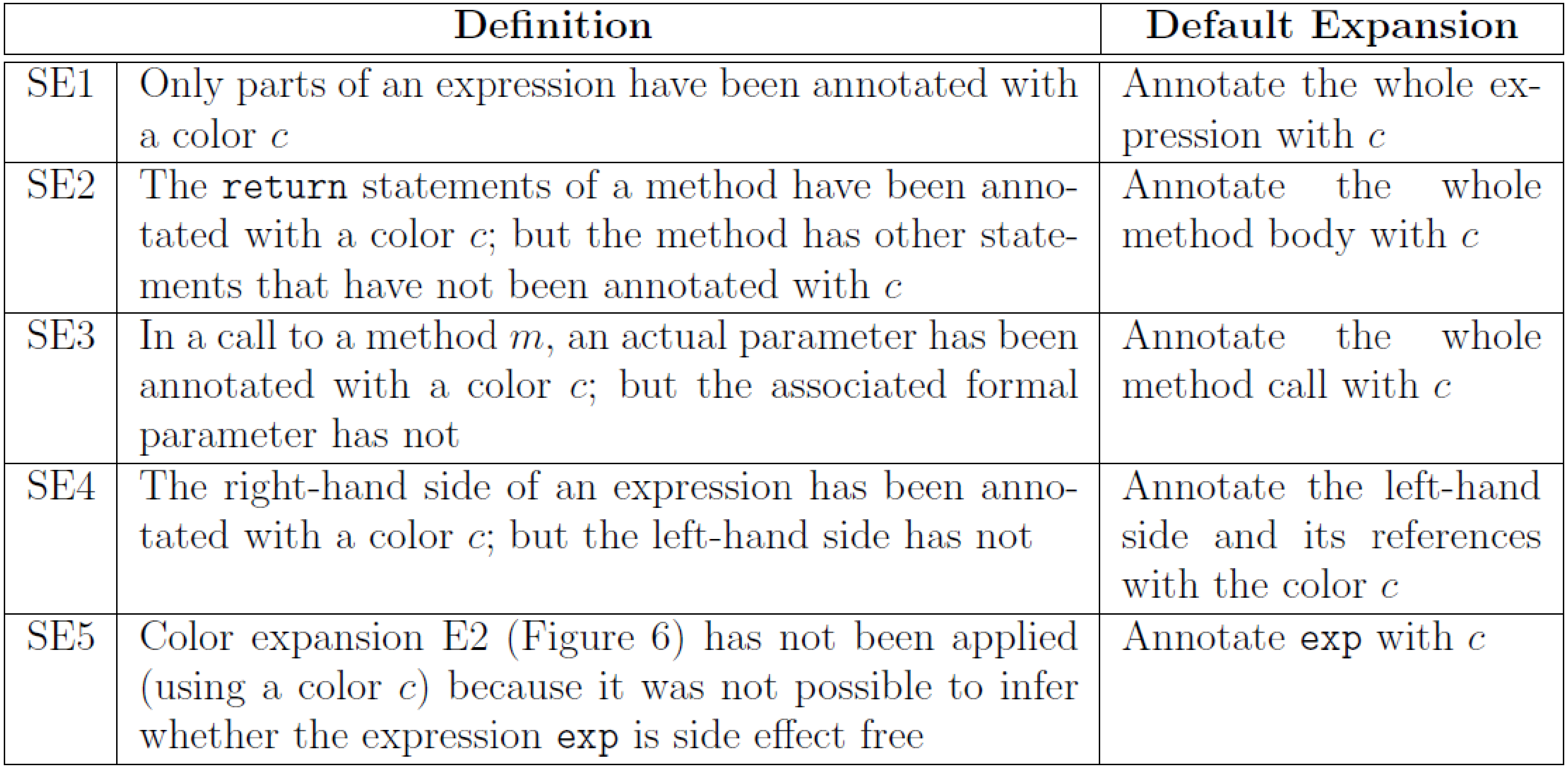
side effects

```
if (bar()) {
   stm;
}
x= y; // bar updates y
```

```
if (bar()) {
   stm;
}
x= y; // bar updates y
```

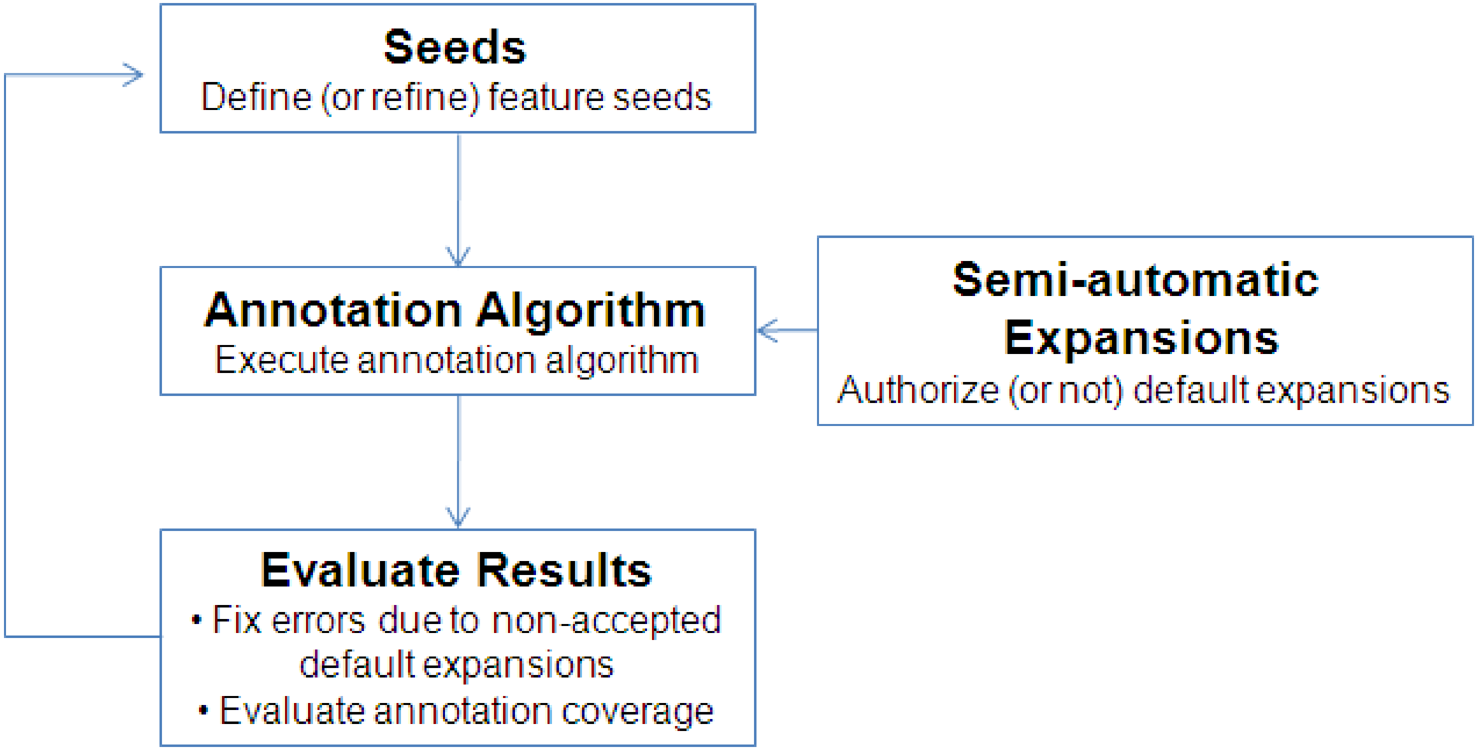
+ warning

Semi-Automatic Expansion Rules



Definition		Default Expansion
SE1	Only parts of an expression have been annotated with a color c	Annotate the whole expression with c
SE2	The `return` statements of a method have been annotated with a color c; but the method has other statements that have not been annotated with c	Annotate the whole method body with c
SE3	In a call to a method m, an actual parameter has been annotated with a color c; but the associated formal parameter has not	Annotate the whole method call with c
SE4	The right-hand side of an expression has been annotated with a color c; but the left-hand side has not	Annotate the left-hand side and its references with the color c
SE5	Color expansion E2 (Figure 6) has not been applied (using a color c) because it was not possible to infer whether the expression `exp` is side effect free	Annotate `exp` with c

Extraction Process



Seeds

Define (or refine) feature seeds

Annotation Algorithm

Execute annotation algorithm

Semi-automatic Expansions

Authorize (or not) default expansions

Evaluate Results

- Fix errors due to non-accepted default expansions
- Evaluate annotation coverage

Evaluation

- Three systems:
  - ArgoUML
  - JFreeChart
  - Prevayler

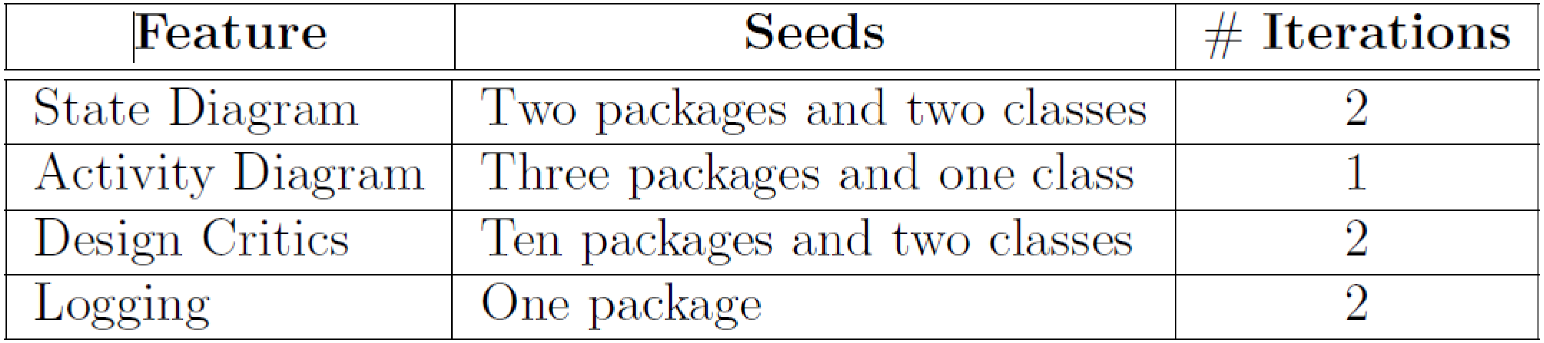
ArgoUML Example

- Manual extraction:
  - Developer with no knowledge about our algorithm
  - Using conditional compilation; then imported to CIDE
  - Public at: http://argouml-spl.tigris.org

Features	KLOC
State Diagrams	3.9
Activity Diagrams	2.2
Design Critics	16.3
Logging	2.5
Total	24.9

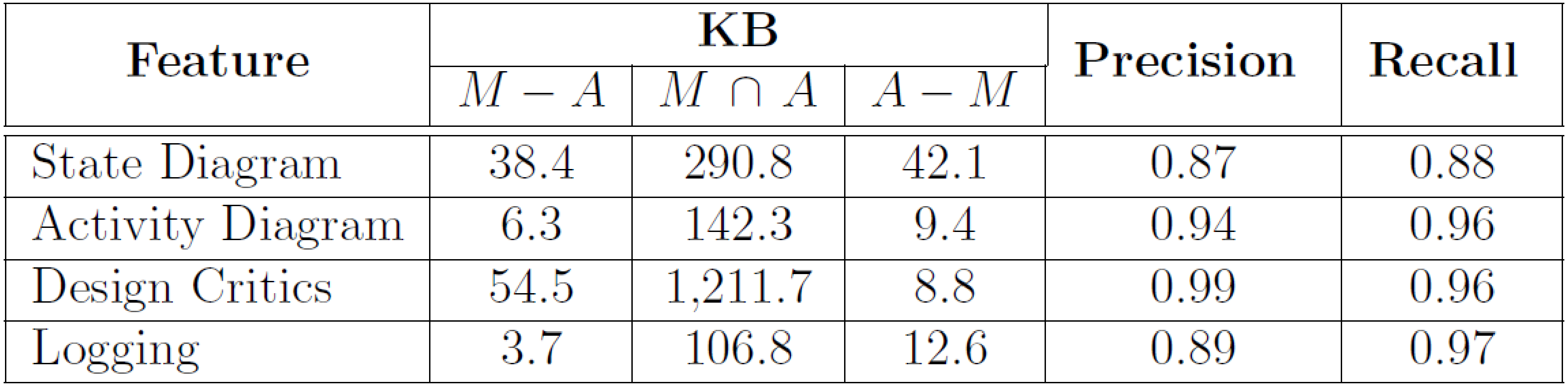
ArgoUML

- Semi-automatic extraction using CIDE+:



Feature	Seeds	# Iterations
State Diagram	Two packages and two classes	2
Activity Diagram	Three packages and one class	1
Design Critics	Ten packages and two classes	2
Logging	One package	2

ArgoUML Results

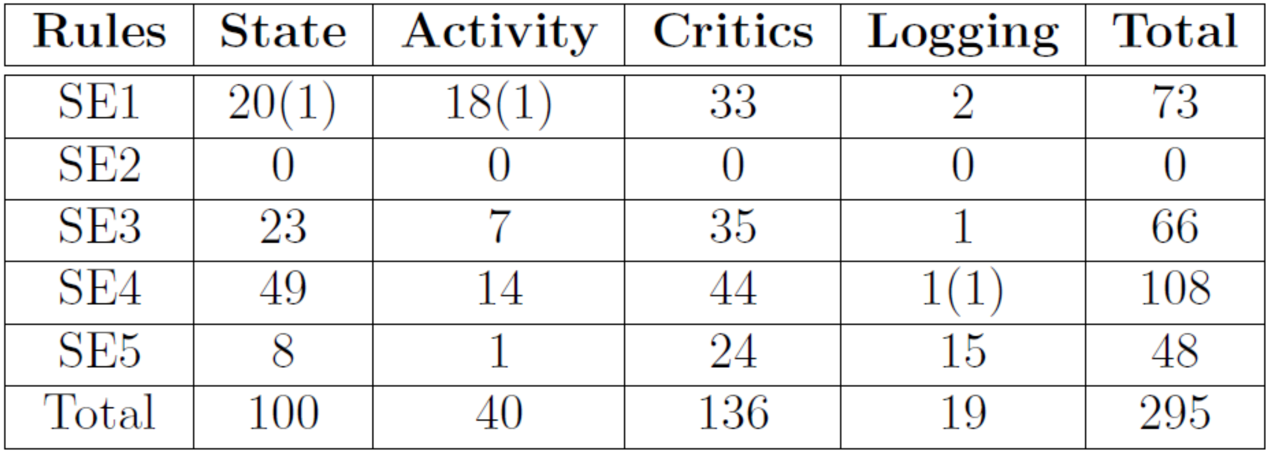


Feature	KB			Precision	Recall
	$M - A$	$M \cap A$	$A - M$		
State Diagram	38.4	290.8	42.1	0.87	0.88
Activity Diagram	6.3	142.3	9.4	0.94	0.96
Design Critics	54.5	1,211.7	8.8	0.99	0.96
Logging	3.7	106.8	12.6	0.89	0.97

(M= Manual extraction; A= semi-automatic extraction)

- recal ≤ 100%:
  - Limitations of the defined seeds in reaching all manual marked code
  - Example: XML parser that process a ToDo list
- precision ≤ 100%:
  - In many parts of the code the developer in charge of the manual extraction has not expanded an annotation to its enclosing context

ArgoUML: Semi-automatic Exp.



Rules	State	Activity	Critics	Logging	Total
SE1	20(1)	18(1)	33	2	73
SE2	0	0	0	0	0
SE3	23	7	35	1	66
SE4	49	14	44	1(1)	108
SE5	8	1	24	15	48
Total	100	40	136	19	295

- In just three cases the developer has not accepted the default actions associated to the proposed semi-automatic expansions

Non-accepted Default Actions

```
cls= org.apache.log4j.Logger.class;
....
cls= Class.forName("...");
```

The RHS of an expression has been annotated with a color c; but the LHS has not

Non-accepted Default Actions

```
cls= org.apache.log4j.Logger.class;
....
cls= Class.forName("...");
```

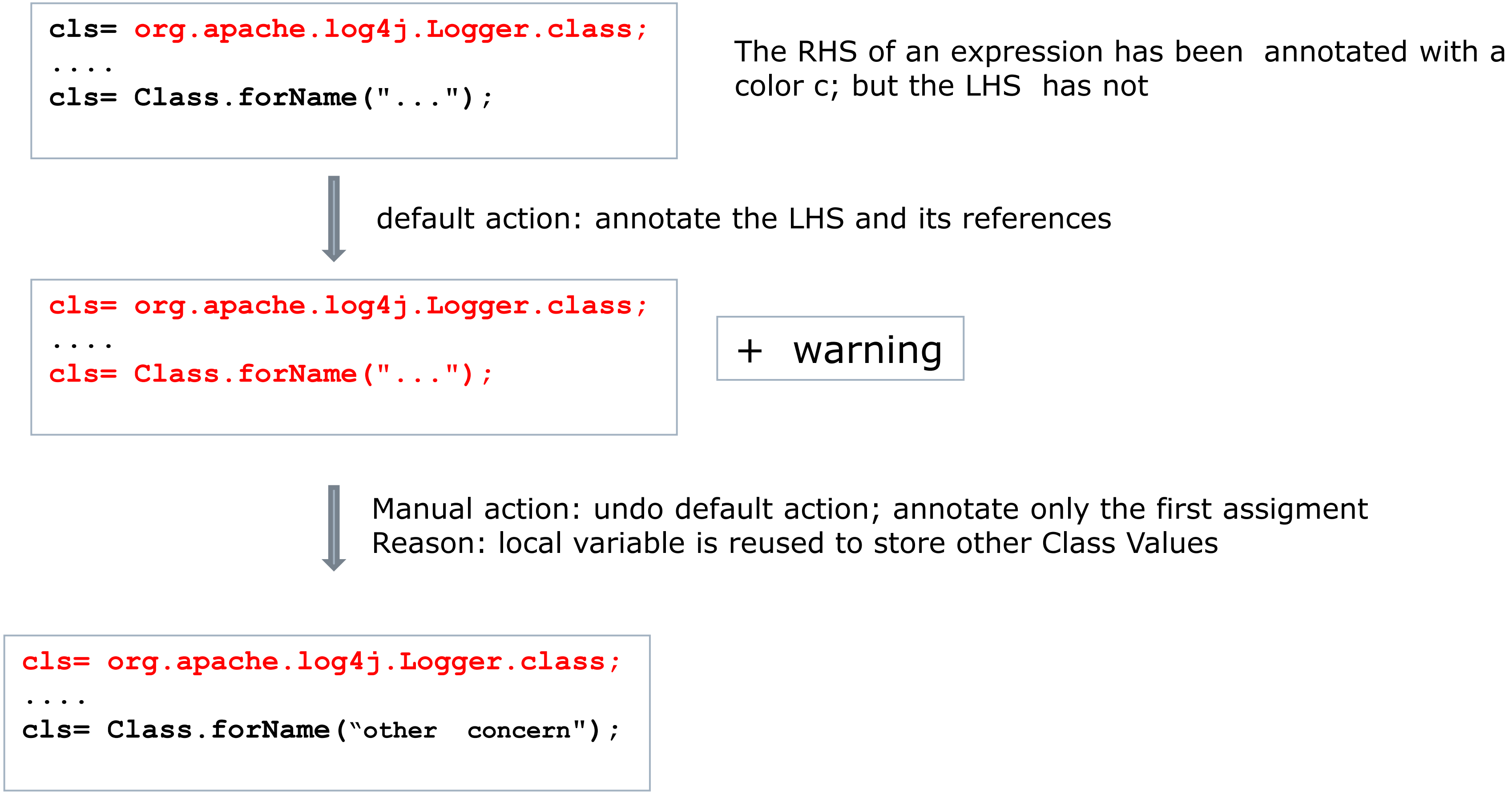
The RHS of an expression has been annotated with a color c; but the LHS has not

default action: annotate the LHS and its references

```
cls= org.apache.log4j.Logger.class;
....
cls= Class.forName(“other  concern");
```

+ warning

Non-accepted Default Actions



```
cls= org.apache.log4j.Logger.class;
....
cls= Class.forName("...");
```

The RHS of an expression has been annotated with a color c; but the LHS has not

default action: annotate the LHS and its references

```
cls= org.apache.log4j.Logger.class;
....
cls= Class.forName("...");
```

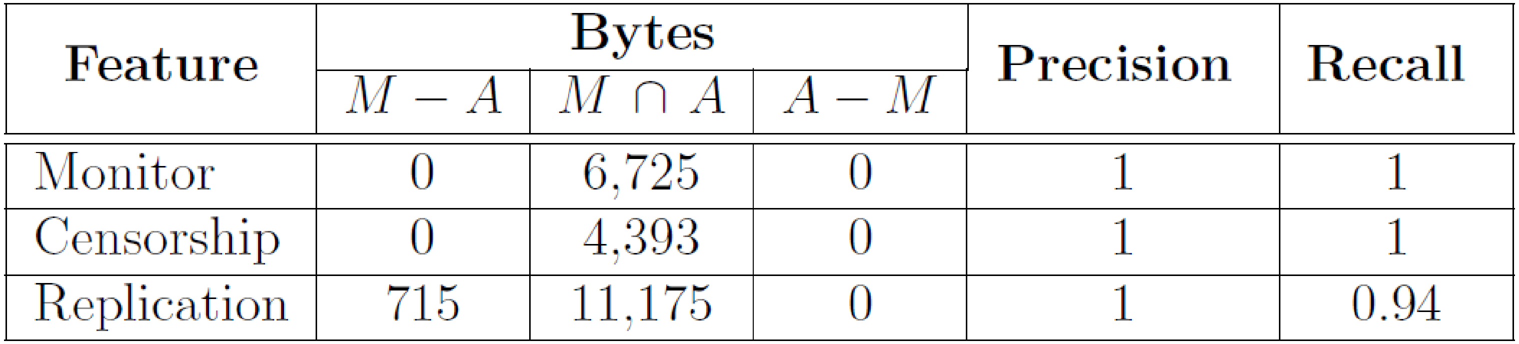
+ warning

Manual action: undo default action; annotate only the first assigment
Reason: local variable is reused to store other Class Values

```
cls= org.apache.log4j.Logger.class;
....
cls= Class.forName(“other  concern");
```

Prevayler

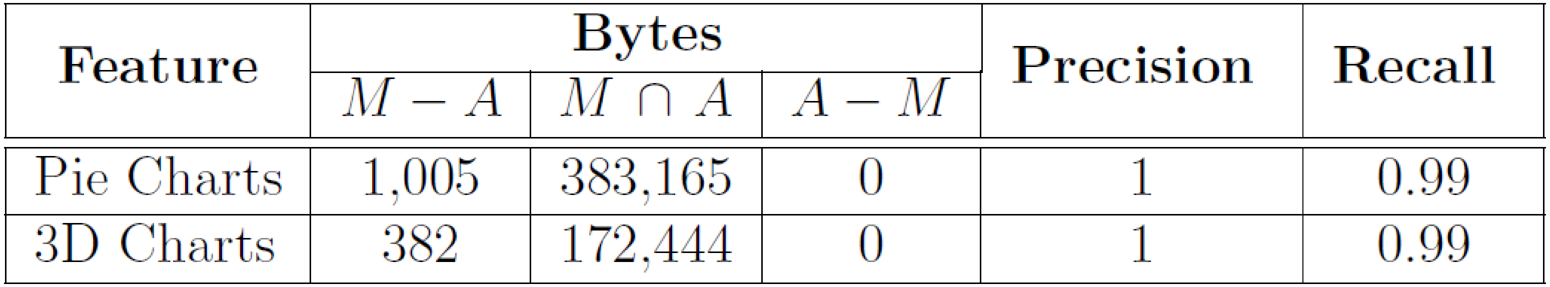
- Similar results: recall, precision, semi-automatic expansions



Feature	Bytes			Precision	Recall
	$M - A$	$M \cap A$	$A - M$		
Monitor	0	6,725	0	1	1
Censorship	0	4,393	0	1	1
Replication	715	11,175	0	1	0.94

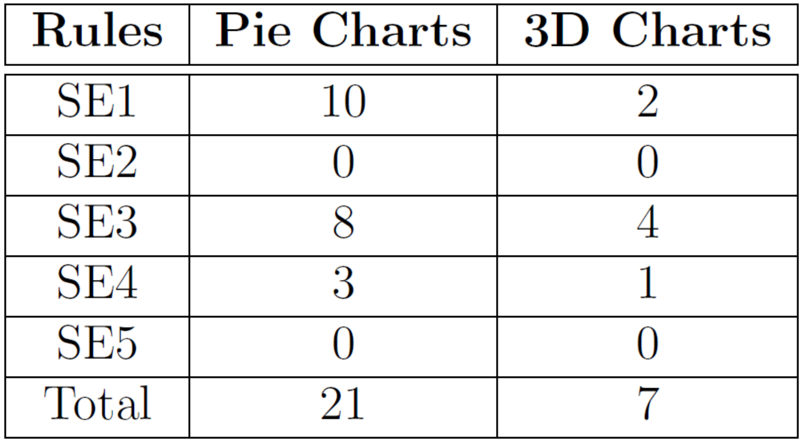
(M= Manual extraction; A= semi-automatic extraction)

JFreeChart



Feature	Bytes			Precision	Recall
	$M - A$	$M \cap A$	$A - M$		
Pie Charts	1,005	383,165	0	1	0.99
3D Charts	382	172,444	0	1	0.99

(M= Manual extraction; A= semi-automatic extraction)



Rules	Pie Charts	3D Charts
SE1	10	2
SE2	0	0
SE3	8	4
SE4	3	1
SE5	0	0
Total	21	7

Semi-automatic expansions (all default actions have been accepted)

Lessons Learned

- SPL extraction is a time-consuming and complex task
- CIDE: promising tool to extract “real” SPLs
- CIDE+: accelerates SPL extraction using CIDE
- However:
  - Developers should be familiar with the target system
  - Developers should carefully select the feature seeds
  - Number of semi-automatic expansions is significant

Related Tools

- Compositional-based approaches (e.g. aspects)
  - Example: AOP-Migrator
  - Do not scale to real, complex SPLs
- Annotation-based approaches (e.g. preprocessors)
  - No tools!

Conclusions

- CIDE+ has been successfully applied in three non-trivial systems

- CIDE+ provides degrees of automation well above existent tools
  - Particularly, when compared with tools based on aspects

- More details, including source code at:
  - www.dcc.ufmg.br/~mtov/cideplus

Thanks