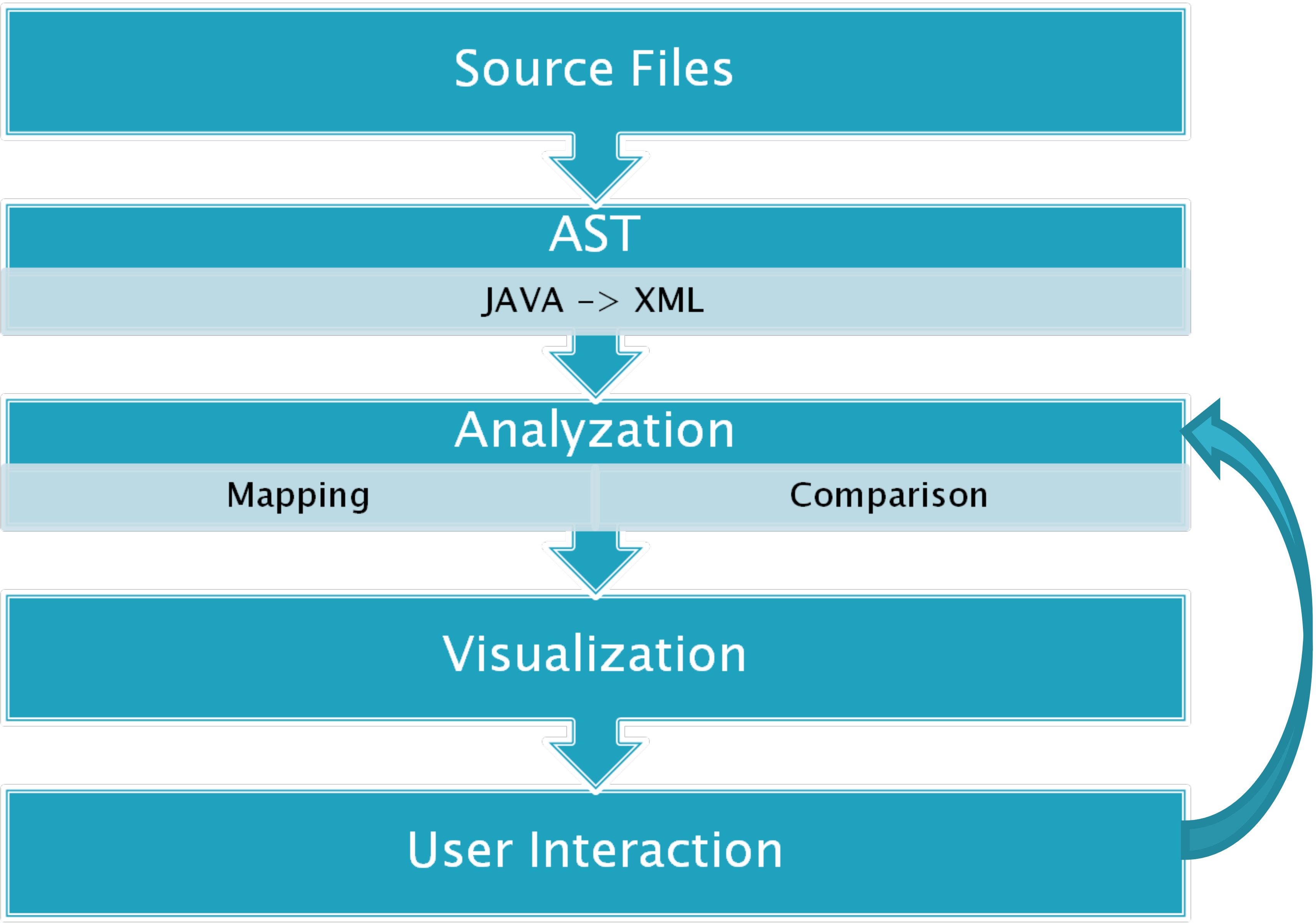
Software Difference Analyzation Tool

Rolf Biehn

Why Abstract Syntax Trees?

- Use an Abstract Syntax Tree(AST) and structurally compare code
    - Reduce noise (such as renamed variables, method declaration order, etc..)
    - Cross-language comparisons become possible
    - Better visualizations possible
    - Improved Navigation



Source Files

AST

JAVA -> XML

Analyzation

Mapping

Comparison

Visualization

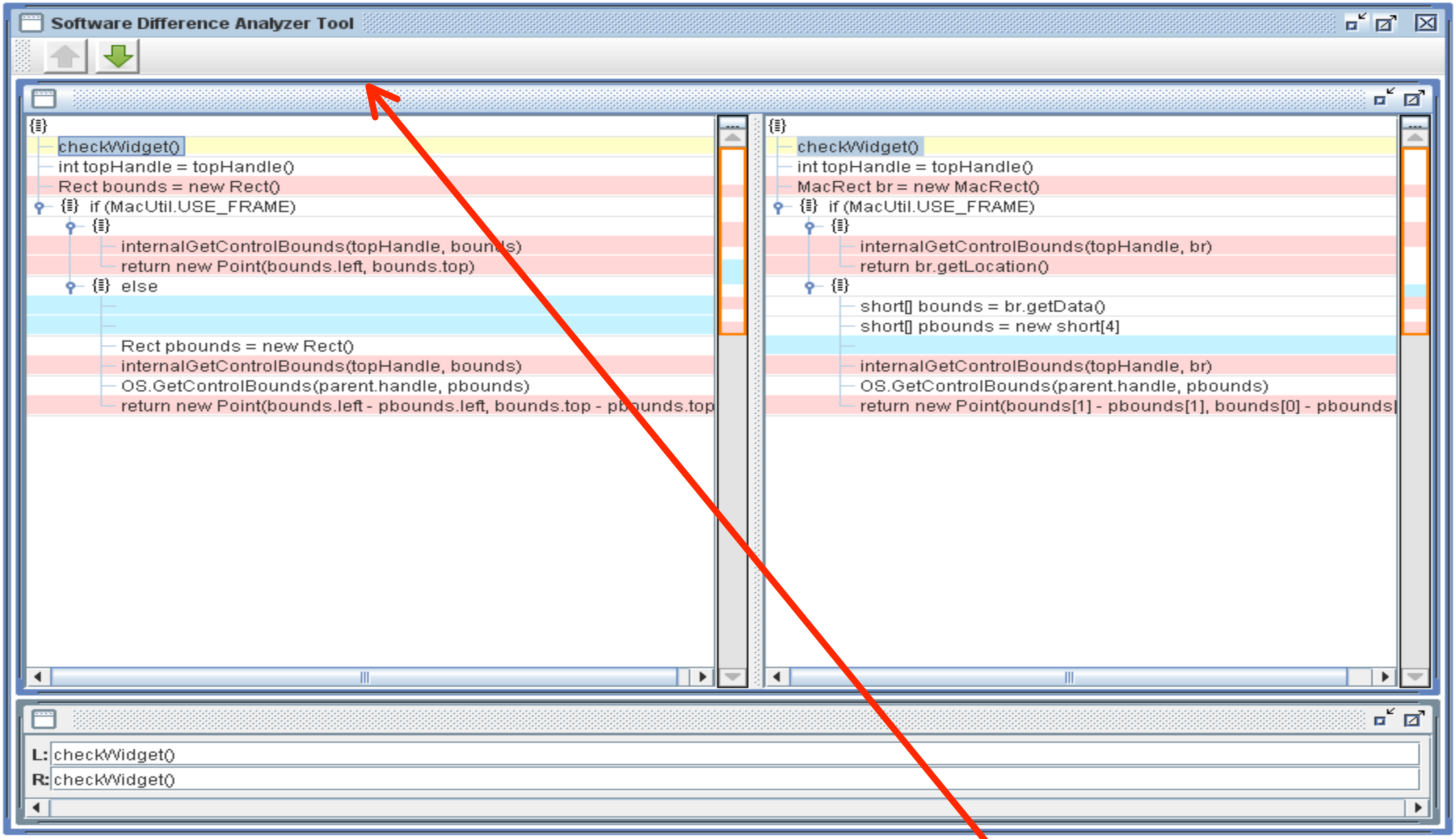
User Interaction

AST & Analysis

- Created my own (partial) AST model
- AST model serializer/deserializer JAVAXML
- Redefined JAVAXML format slightly (MethodName, ClassName, VariableName attributes → name)
- Simple Comparison + manual intervention

AST & Analysis (Problems)

- JAVA XML scalability issues (5K Lines of code -> 800K of memory)
- Limited number of AST nodes supported
- Need to redo my AST object model



```
{≡}
checkWidget()
int topHandle = topHandle()
Rect bounds = new Rect()
{≡} if (MacUtil.USE_FRAME)
    {≡}
        internalGetControlBounds(topHandle, bounds)
        return new Point(bounds.left, bounds.top)
    {≡} else


        Rect pbounds = new Rect()
        internalGetControlBounds(topHandle, bounds)
        OS.GetControlBounds(parent.handle, pbounds)
        return new Point(bounds.left - pbounds.left, bounds.top - pbounds.top
```

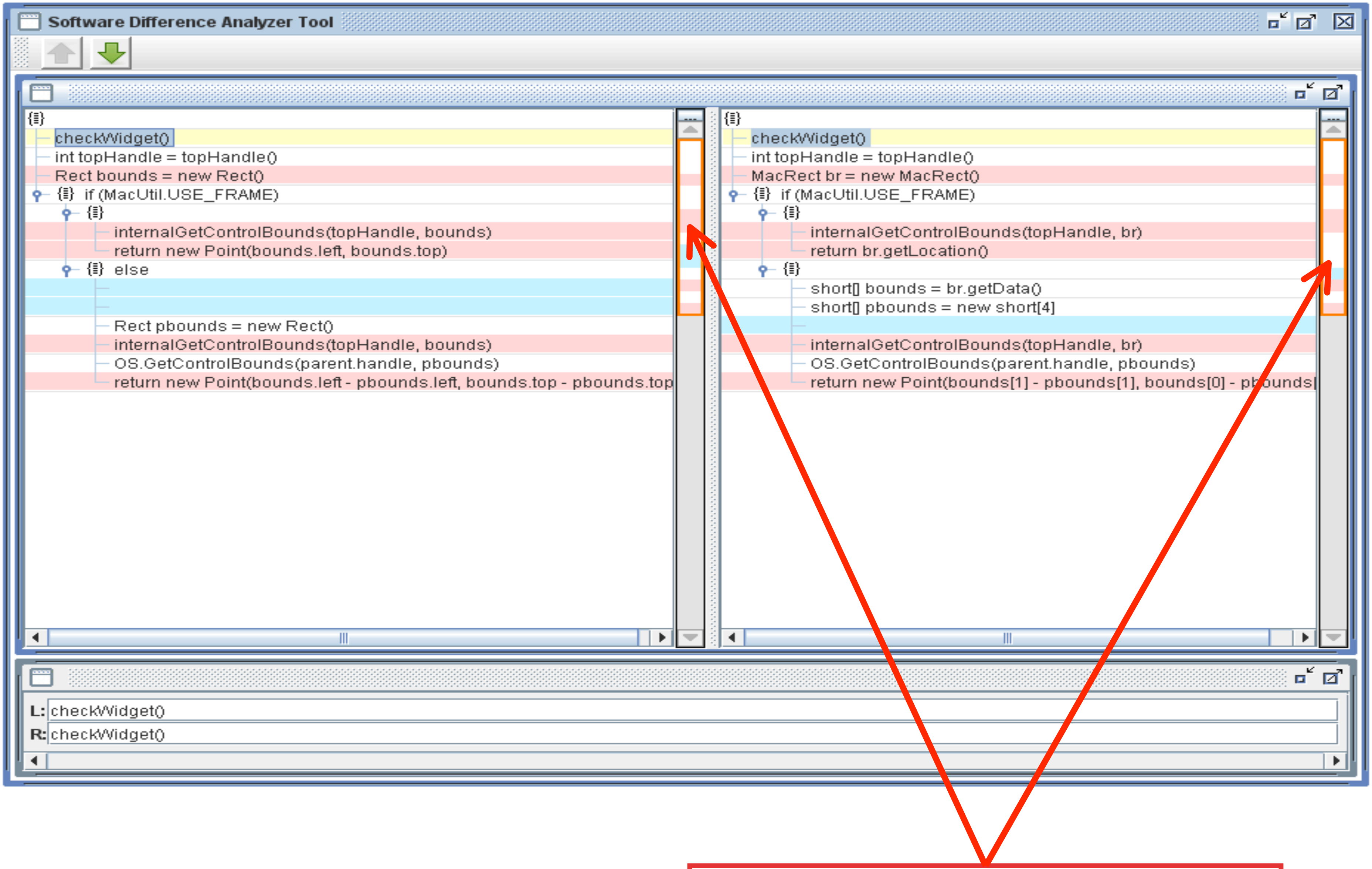
```
{≡}
checkWidget()
int topHandle = topHandle()
MacRect br = new MacRect()
{≡} if (MacUtil.USE_FRAME)
    {≡}
        internalGetControlBounds(topHandle, br)
        return br.getLocation()
    {≡}
        short[] bounds = br.getData()
        short[] pbounds = new short[4]

        internalGetControlBounds(topHandle, br)
        OS.GetControlBounds(parent.handle, pbounds)
        return new Point(bounds[1] - pbounds[1], bounds[0] - pbounds[
```

L: checkWidget()

R: checkWidget()

Method View



Software Difference Analyzer Tool

```
{≡}
checkWidget()
int topHandle = topHandle()
Rect bounds = new Rect()
{≡} if (MacUtil.USE_FRAME)
    {≡}
        internalGetControlBounds(topHandle, bounds)
        return new Point(bounds.left, bounds.top)
    {≡} else


        Rect pbounds = new Rect()
        internalGetControlBounds(topHandle, bounds)
        OS.GetControlBounds(parent.handle, pbounds)
        return new Point(bounds.left - pbounds.left, bounds.top - pbounds.top
```

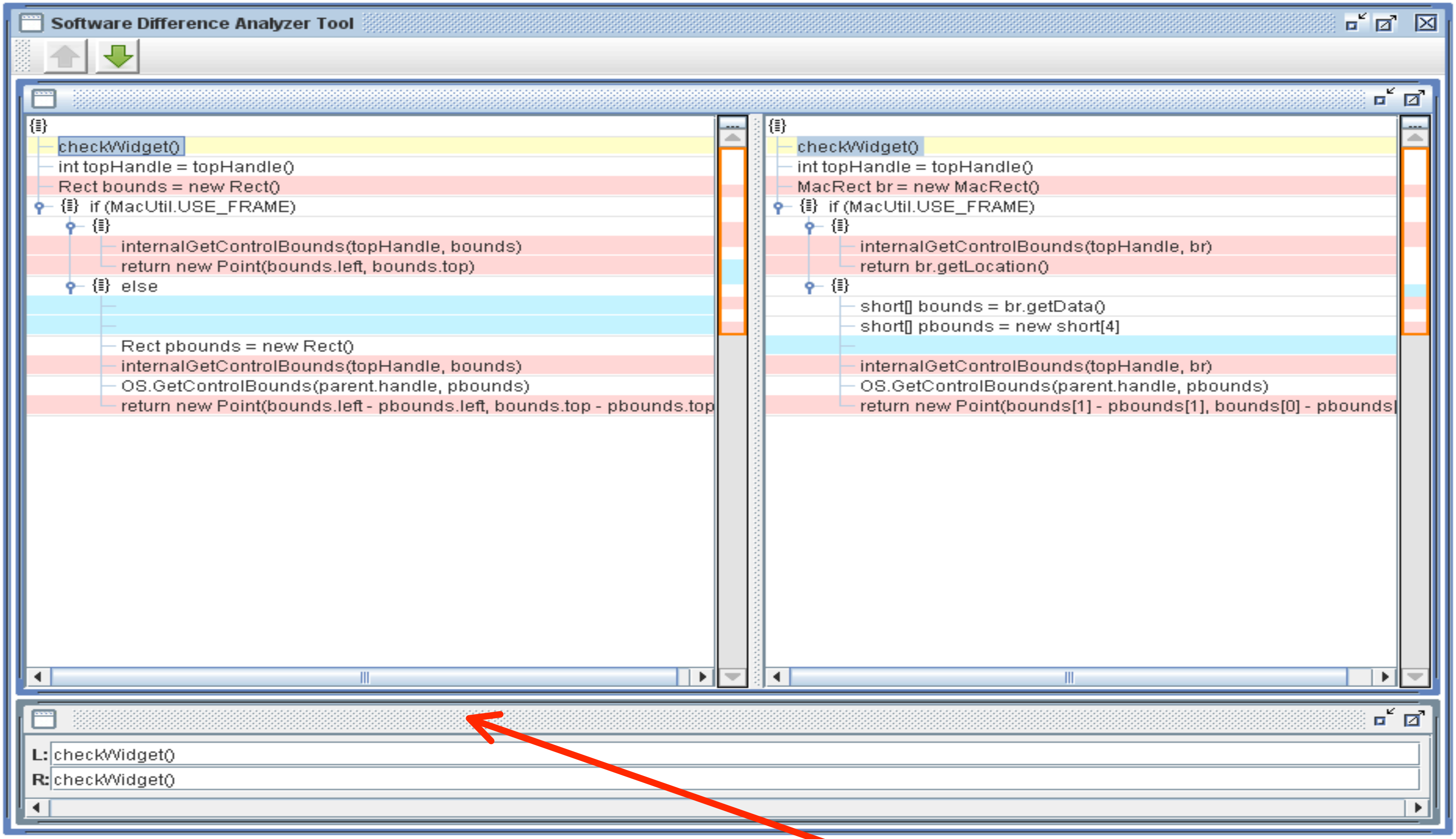
```
{≡}
checkWidget()
int topHandle = topHandle()
MacRect br = new MacRect()
{≡} if (MacUtil.USE_FRAME)
    {≡}
        internalGetControlBounds(topHandle, br)
        return br.getLocation()
    {≡}
        short[] bounds = br.getData()
        short[] pbounds = new short[4]

        internalGetControlBounds(topHandle, br)
        OS.GetControlBounds(parent.handle, pbounds)
        return new Point(bounds[1] - pbounds[1], bounds[0] - pbounds|
```

L: checkWidget()

R: checkWidget()

Mini-Map



```
{≡}
 checkWidget()
 int topHandle = topHandle()
 Rect bounds = new Rect()
 {≡} if (MacUtil.USE_FRAME)
   {≡}
     internalGetControlBounds(topHandle, bounds)
     return new Point(bounds.left, bounds.top)
   {≡} else


     Rect pbounds = new Rect()
     internalGetControlBounds(topHandle, bounds)
     OS.GetControlBounds(parent.handle, pbounds)
     return new Point(bounds.left - pbounds.left, bounds.top - pbounds.top
```

```
{≡}
 checkWidget()
 int topHandle = topHandle()
 MacRect br = new MacRect()
 {≡} if (MacUtil.USE_FRAME)
   {≡}
     internalGetControlBounds(topHandle, br)
     return br.getLocation()
   {≡}
     short[] bounds = br.getData()
     short[] pbounds = new short[4]

     internalGetControlBounds(topHandle, br)
     OS.GetControlBounds(parent.handle, pbounds)
     return new Point(bounds[1] - pbounds[1], bounds[0] - pbounds|
```

L: checkWidget()

R: checkWidget()

Detailed View

Adapt the AST Model

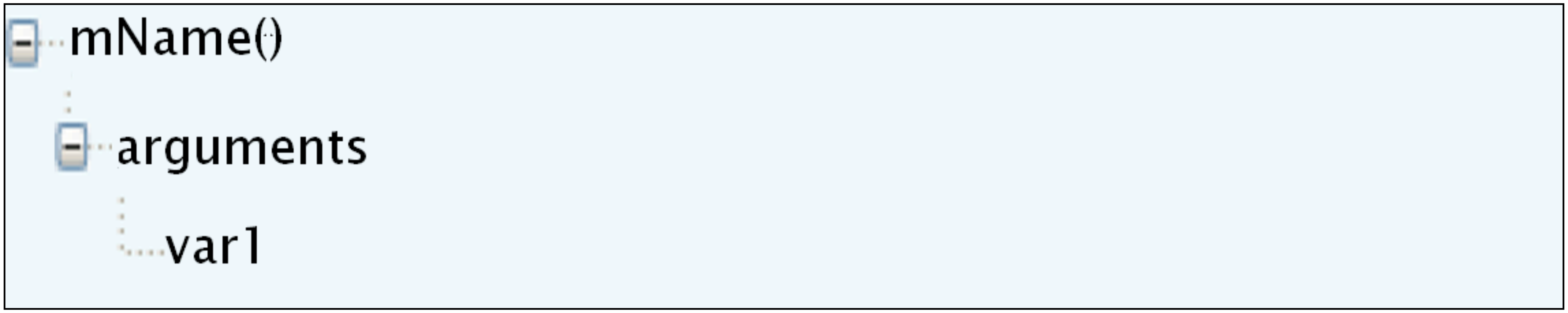
```
<methodCall name=“mName">
    <meta name="diff" />
    <arguments>
         <var-ref name=“var1" />
```

- In JAVA this would be “mName(var1);”

Adapt the AST Model

```
<methodCall name=“mName">
    <meta name="diff" />
    <arguments>
        <var-ref name=“var1" />
```

- We don’t want to Visualize the AST Model.



```
⊟ mName()
  ⊟ arguments
      var1
```

Flatten the Tree

```
<methodCall name=“mName">
   <meta name="diff" />
   <arguments>
      <var-ref name=“var1" />
```

- Therefore, flatten the list.

```
{ <DIFF>, <“mName”>, </DIFF>, <“(“>, <“var1”>,
<“)”> }
```

Mini-Map Demo

- Markers (Red, Blue, Purple)
- Collapsible (purple – updates both, efficient)
- Left click moves to spot
- ViewMode (anchor, length, current size)
- Page up / down buttons
- %Width
- Locked Mode
- Resize Mode

Mini-Map

- The Good
  - Can represent a large number of lines
  - Interaction is easy once learned
- The Bad
  - Not intuitive?
  - Resize mode needs help
- The Future
  - Further Optimizations possible
  - User Studies
  - Clean up Interfaces for library re-use

Detailed View Demo

Final Conclusions

- Mini-Map can be used anywhere with a pixel map + line concept (i.e. SeeSoft)
- Not really leveraging AST comparison at this point
- Piece of a puzzle – long road ahead including:
  - More support of JAVA & C# AST nodes
  - Better comparison algorithms
  - Able to compare and visualize classes, packages, etc..
  - More interaction {filters, go to, multiple views, etc..}

Questions?