

# **FLAT<sup>3</sup>: Feature Location & Textual Tracing Tool**

Trevor Savage, Meghan Revelle, Denys Poshyvanyk

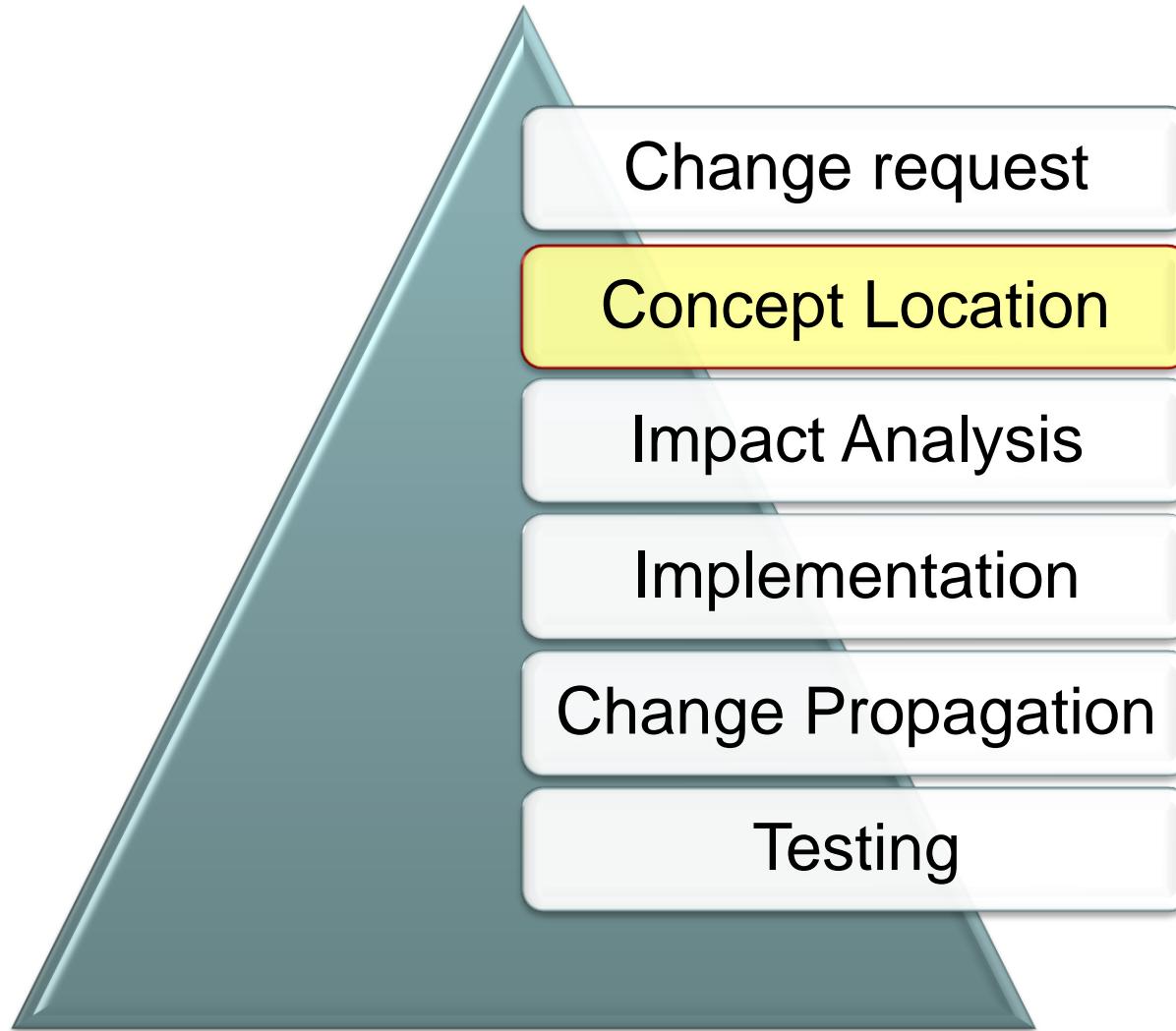
SEMERU Group @ William and Mary



# Addressed Problem

- The software developer has to maintain large software systems with:
  - Little or no domain knowledge
  - Absence of the original developer
  - Badly organized, missing, or out of date documentation

# Concept Location in Source Code



# “Finding a Needle in a Haystack”

```

*** BTree::Search const TBB::BTree& BTree::Search()
{
    Node* current = root;
    while (current != NULL)
    {
        if (key == current->key)
            return current;
        else if (key < current->key)
            current = current->left;
        else
            current = current->right;
    }
    return NULL;
}

BTree::BTree(BTree left, BTree right)
{
    parent = parent;
    keyType key;
    _left(left);
    _right(right);
    _parent(parent);
    _item(item());
}

BTree::BTree(const BTree& parent)
{
    parent = parent;
    _item(item());
}

void BTree::setKey(keyType key)
{
    _item.first = key;
}

keyType BTree::getFirst()
{
    return _item.first;
}

void BTree::setValue(itemType value)
{
    _item.second = value;
}

std::ostream& operator<< (std::ostream& os, const BTree& bt)
{
    os << bt._item.first << " " << bt._item.second << "\n";
    return os;
}

pair<BTree*, bool> BTree::Find(keyType key)
{
    search tree for key before performing
    // if key found, return false
    // if key found, return parent where should go, false
    // if found, add node to with key, true
}

```

```

class BinarySearchTree implements Tree
{
    private Node root;
    private int size;
    private int height;

    public BinarySearchTree()
    {
        root = null;
        size = 0;
        height = 0;
    }

    public int size()
    {
        return size;
    }

    public int height()
    {
        return height;
    }

    public void insert(int item)
    {
        if (root == null)
            root = new Node(item);
        else
            insert(root, item);
    }

    private void insert(Node node, int item)
    {
        if (item < node.item)
            if (node.left == null)
                node.left = new Node(item);
            else
                insert(node.left, item);
        else
            if (node.right == null)
                node.right = new Node(item);
            else
                insert(node.right, item);
    }

    public Node search(int item)
    {
        return search(root, item);
    }

    private Node search(Node node, int item)
    {
        if (node == null)
            return null;
        else
            if (item == node.item)
                return node;
            else
                if (item < node.item)
                    return search(node.left, item);
                else
                    return search(node.right, item);
    }

    public int min()
    {
        return min(root).item;
    }

    private Node min(Node node)
    {
        if (node.left == null)
            return node;
        else
            return min(node.left);
    }

    public int max()
    {
        return max(root).item;
    }

    private Node max(Node node)
    {
        if (node.right == null)
            return node;
        else
            return max(node.right);
    }

    public int predecessor(int item)
    {
        return predecessor(root, item);
    }

    private int predecessor(Node node, int item)
    {
        if (node == null)
            return Integer.MAX_VALUE;
        else
            if (item < node.item)
                return predecessor(node.left, item);
            else
                if (node.left != null)
                    return max(node.left);
                else
                    return predecessor(node.parent, item);
    }

    public int successor(int item)
    {
        return successor(root, item);
    }

    private int successor(Node node, int item)
    {
        if (node == null)
            return Integer.MIN_VALUE;
        else
            if (item > node.item)
                return successor(node.right, item);
            else
                if (node.right != null)
                    return min(node.right);
                else
                    return successor(node.parent, item);
    }

    public void print()
    {
        print(root);
    }

    private void print(Node node)
    {
        if (node == null)
            return;
        else
            System.out.println(node.item);
        print(node.left);
        print(node.right);
    }
}

```

```

// binary search tree implementation
* Data Structures
*
* Node<T>
* Binode<T>
* BTree<T>
* BTreeIter<T>
* BTreeConstIter<T>
* BTreeConstIter<T> const

* Node<T>
Binode<T>::Binode()
Binode<T>::Binode(string name)
Binode<T>::Binode(string name, utility::pair<Binode<T>, Binode<T> > parent, pair<Binode<T>, ValueType> item)
Binode<T>::Binode(string name, utility::pair<Binode<T>, Binode<T> > parent, pair<Binode<T>, ValueType> item,
                  _leftItem, _rightItem, _parentItem, _itemItem) {}

KeyType BTree<T>::getByKey(KeyType key)
{
    return _tree.first;
}

void BTree<T>::setByKey(KeyType key)
{
    _tree.first = key;
}

KeyType BTree<T>::getValByName()
{
    return _tree.second;
}

void BTree<T>::setValByName(ValueType value)
{
    _tree.second = value;
}

std::ostream &operator<<(std::ostream &os, const BTree<T> &t)
{
    os << "{'" << t._tree.first << "'": {" << t._tree.second << "'"} }";
    return os;
}

pair<Binode<T>, bool> BST::find(keyType key)
{
    if (_root == NULL)
        return {NULL, false};
    if (key < _root->_itemItem)
        return find(_root->_leftItem, key);
    if (key > _root->_itemItem)
        return find(_root->_rightItem, key);
    return { _root, true };
}

pair<Binode<T>, bool> BST::find(keyType key) const
{
    if (_root == NULL)
        return {NULL, false};
    if (key < _root->_itemItem)
        return find(_root->_leftItem, key);
    if (key > _root->_itemItem)
        return find(_root->_rightItem, key);
    return { _root, true };
}

utility::pair<Binode<T>, ValueType> BST::findMin() const
{
    if (_root == NULL)
        return {NULL, -1};
    Binode<T> *current = _root;
    while (current->_leftItem != NULL)
        current = current->_leftItem;
    return {current, current->_itemItem};
}

utility::pair<Binode<T>, ValueType> BST::findMax() const
{
    if (_root == NULL)
        return {NULL, -1};
    Binode<T> *current = _root;
    while (current->_rightItem != NULL)
        current = current->_rightItem;
    return {current, current->_itemItem};
}

utility::pair<Binode<T>, ValueType> BST::findSuccessor() const
{
    if (_root == NULL)
        return {NULL, -1};
    if (_root->_rightItem == NULL)
        return { _root, _root->_itemItem };
    Binode<T> *current = _root->_rightItem;
    while (current->_leftItem != NULL)
        current = current->_leftItem;
    return {current, current->_itemItem};
}

utility::pair<Binode<T>, ValueType> BST::findPredecessor() const
{
    if (_root == NULL)
        return {NULL, -1};
    if (_root->_leftItem == NULL)
        return { _root, _root->_itemItem };
    Binode<T> *current = _root->_leftItem;
    while (current->_rightItem != NULL)
        current = current->_rightItem;
    return {current, current->_itemItem};
}

utility::pair<Binode<T>, ValueType> BST::findSuccessor(Binode<T> *node) const
{
    if (node == NULL)
        return {NULL, -1};
    if (node->_rightItem == NULL)
        return { node, node->_itemItem };
    Binode<T> *current = node->_rightItem;
    while (current->_leftItem != NULL)
        current = current->_leftItem;
    return {current, current->_itemItem};
}

utility::pair<Binode<T>, ValueType> BST::findPredecessor(Binode<T> *node) const
{
    if (node == NULL)
        return {NULL, -1};
    if (node->_leftItem == NULL)
        return { node, node->_itemItem };
    Binode<T> *current = node->_leftItem;
    while (current->_rightItem != NULL)
        current = current->_rightItem;
    return {current, current->_itemItem};
}

utility::pair<Binode<T>, ValueType> BST::findSuccessor(Binode<T> *node)
{
    if (node == NULL)
        return {NULL, -1};
    if (node->_rightItem == NULL)
        return { node, node->_itemItem };
    Binode<T> *current = node->_rightItem;
    while (current->_leftItem != NULL)
        current = current->_leftItem;
    return {current, current->_itemItem};
}

utility::pair<Binode<T>, ValueType> BST::findPredecessor(Binode<T> *node)
{
    if (node == NULL)
        return {NULL, -1};
    if (node->_leftItem == NULL)
        return { node, node->_itemItem };
    Binode<T> *current = node->_leftItem;
    while (current->_rightItem != NULL)
        current = current->_rightItem;
    return {current, current->_itemItem};
}

```

```

binary search tree implementation
* file: bst.h
* author: www.dreamincode.net
* date: 2011-07-24

include <fstream>
include <iostream>
include <algorithm>
include <utility>
include <vector>
using std::ios
using std::cout
using std::endl
using std::pair
using std::string

class BTHnode
{
    private BTHnode* left, BTHnode* right;
    private KeyType item;
    private ValueType parent;
    private ValueType item;
    private ValueType left;
    private ValueType right;
    private ValueType parent;
    private ValueType item;

    public BTHnode( KeyType key, ValueType value )
    {
        this.item = key;
        this.parent = value;
        this.left = NULL;
        this.right = NULL;
    }

    public BTHnode( KeyType key )
    {
        this.item = key;
        this.parent = NULL;
        this.left = NULL;
        this.right = NULL;
    }

    public KeyType getkey()
    {
        return this.item;
    }

    public ValueType getvalue()
    {
        return this.parent;
    }

    public void setvalue( ValueType value )
    {
        this.parent = value;
    }

    public operator << ( std::ostream& os, const BTHnode* v ) helps
    {
        os << v->item;
        os << " ";
        os << v->parent;
        os << endl;
        os << v->left;
        os << v->right;
        os << endl;
    }
}

pair(BTHnode*, bool) BST::find( KeyType key )
{
    if ( this == NULL ) return { NULL, false };
    if ( key < this->item ) return find( key );
    if ( key > this->item ) return { NULL, false };
    else return { this, true };
}

pair(BTHnode*, bool) BST::find( KeyType key )
{
    if ( this == NULL ) return { NULL, false };
    if ( key < this->item ) return find( key );
    if ( key > this->item ) return { NULL, false };
    else return { this, false };
}

```

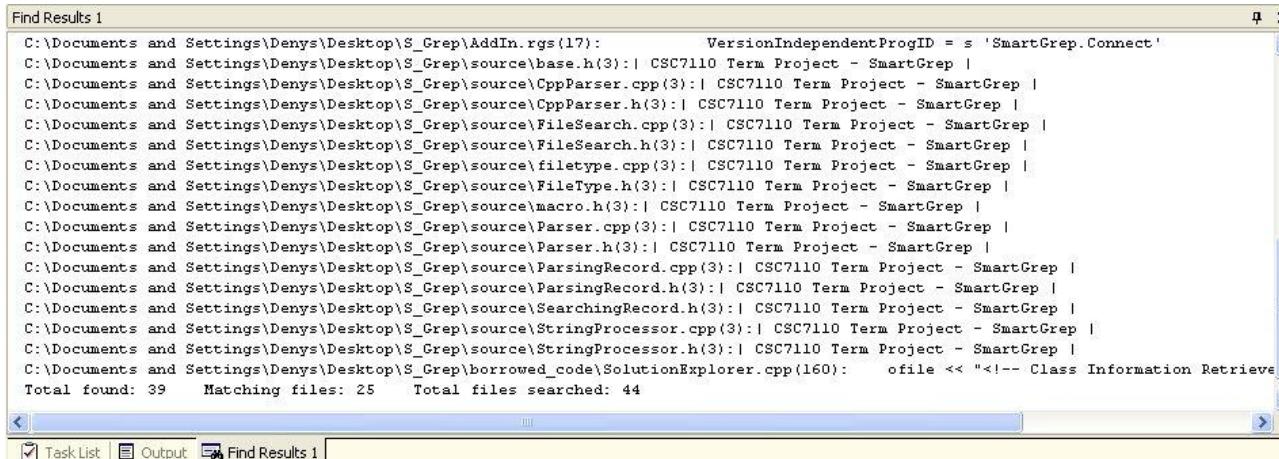
# Concept Location with Regular Expressions

```
BSTnode& BST::insert(KeyType key, ValueType value)
// lookup key, create key/value if not present
// return pointer to node with key
{
    BSTnode& n = getNode(key);
    n._item.second = value;
    return n;
}

BSTnode& BST::getNode(KeyType key)
// lookup key, create key/value if not present
// return pointer to node with key
{
    pair<KeyType, ValueType> item(key, 0); // create item to be inserted

    if (_root == 0)
    {
        BSTnode* new_node = new BSTnode(0, 0, 0, item);
        _root = new_node;
        return *_root;
    }

    pair<BSTnode*, bool> result = find(key); // search for key
    // location found
    if (result.second) // true, node exists
    {
        return *(result.first);
    }
    else // false, create node
    {
        BSTnode* parent = result.first;
        BSTnode* new_node = new BSTnode(0, 0, parent, item);
        if (!key.parent->getKey())
            parent->left = new_node;
        else
            parent->right = new_node;
        return *new_node;
    }
}
```



# Shortcomings of Static Concept Location

- Highly dependent on naming conventions and the developer's experience to write good queries
- Ignores other existing relationships between software components (such as, dependencies)
- May miss important parts of the source code

# Feature Location with Software Reconnaissance - Dynamic Analysis

[Wilde'92] [Antoniol'06]

## Scenario NOT exercising the feature (trace 1)

```
readAndDispatch -- org.eclipse.swt.widgets.Display
checkDevice -- org.eclipse.swt.widgets.Display
isDisposed -- org.eclipse.swt.graphics.Device
drawMenuBar -- org.eclipse.swt.widgets.Display
runPopups -- org.eclipse.swt.widgets.Display
filterMessage -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Control checkDevice -- org.eclipse.swt.widgets.Display
WM_TIMER -- org.eclipse.swt.widgets.Control isDisposed -- org.eclipse.swt.graphics.Device
windowProc -- org.eclipse.swt.widgets.Display drawMenuBar -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Control runPopups -- org.eclipse.swt.widgets.Display
WM_TIMER -- org.eclipse.swt.widgets.Control filterMessage -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Display windowProc -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Control
readAndDispatch -- org.eclipse.swt.widgets.Display
checkDevice -- org.eclipse.swt.widgets.Display
isDisposed -- org.eclipse.swt.graphics.Device
drawMenuBar -- org.eclipse.swt.widgets.Display
runPopups -- org.eclipse.swt.widgets.Display
runAsyncMessages -- org.eclipse.swt.widgets.Display
removeFirst -- org.eclipse.swt.widgets.Synchronizer
```

## Scenario exercising the feature (trace 2)

```
checkDevice -- org.eclipse.swt.widgets.Display
isDisposed -- org.eclipse.swt.graphics.Device
drawMenuBar -- org.eclipse.swt.widgets.Display
runPopups -- org.eclipse.swt.widgets.Display
runAsyncMessages -- org.eclipse.swt.widgets.Display
removeFirst -- org.eclipse.swt.widgets.Synchronizer
```

# Shortcomings of Dynamic Concept Location

- Execution traces are large even for small systems
- Selecting (ir)relevant scenarios may be difficult
- Filtering the traces is equally problematic - best filtering methods still return hundreds of methods

# Single Trace Information Retrieval (SITIR)

[ASE'07]

## Single Execution Trace

```

readAndDispatch -- org.eclipse.swt.widgets.Display
checkDevice -- org.eclipse.swt.widgets.Display
isDisposed -- org.eclipse.swt.graphics.Device
drawMenuBar -- org.eclipse.swt.widgets.Display
runPopups -- org.eclipse.swt.widgets.Display
filterMessage -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Control
WM_TIMER -- org.eclipse.swt.widgets.Control
windowProc -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Control
WM_TIMER -- org.eclipse.swt.widgets.Control
windowProc -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Control
readAndDispatch -- org.eclipse.swt.widgets.Display
checkDevice -- org.eclipse.swt.widgets.Display
isDisposed -- org.eclipse.swt.graphics.Device
drawMenuBar -- org.eclipse.swt.widgets.Display
runPopups -- org.eclipse.swt.widgets.Display
filterMessage -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Control
WM_TIMER -- org.eclipse.swt.widgets.Control
windowProc -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Control
WM_TIMER -- org.eclipse.swt.widgets.Control
windowProc -- org.eclipse.swt.widgets.Display
windowProc -- org.eclipse.swt.widgets.Control
  
```

The diagram illustrates the process of extracting information from source code. It starts with three identical blocks of C code for a BSTNode class, each containing functions for inserting, getting, and finding key-value pairs. A large blue arrow points from these blocks to a central box labeled "Information Retrieval Engine". Another blue arrow points from the engine box to a table below.

```

BSTnode BST_insert(KeyType key, ValueType value)
{
    if (root == NULL) // if tree is empty
        return pointer to node with key
    else
    {
        BSTnode a = getMode(key);
        if (a->second == value)
            return a;
    }
}

BSTnode BST_getMode(KeyType key)
{
    if (root == NULL) // if tree is empty
        return pointer to node with key
    else
    {
        pair<KeyType,ValueType> item(key, 0); // create item to be inserted
        if (_root == 0)
        {
            BSTnode new_node = new BSTnode(0, 0, item);
            root = new_node;
            return new_node;
        }
        pair<KeyType,ValueType> result(item, 0); // search for key
        if (!result.second) // location found
            return *result.first; // tree node exists
        else
        {
            BSTnode parent = result.first;
            BSTnode new_node = new BSTnode(0, 0, parent->item);
            if (parent->left == new_node)
                parent->left = new_node;
            else
                parent->right = new_node;
            return *new_node;
        }
    }
}

pair<KeyType,ValueType> BST_find(KeyType key)
{
    if (root == NULL) // if tree is empty
        return pair<KeyType,ValueType>(key, 0);
    else
    {
        pair<KeyType,ValueType> item(key, 0); // create item to be inserted
        if (_root == 0)
        {
            BSTnode new_node = new BSTnode(0, 0, item);
            root = new_node;
            return pair<KeyType,ValueType>(key, 0);
        }
        pair<KeyType,ValueType> result(item, 0); // search for key
        if (!result.second) // location found
            return result; // tree node exists
        else
        {
            BSTnode parent = result.first;
            BSTnode new_node = new BSTnode(0, 0, parent->item);
            if (parent->left == new_node)
                parent->left = new_node;
            else
                parent->right = new_node;
            return result;
        }
    }
}
  
```

## Source Code

```

BSTnode BST_insert(KeyType key, ValueType value)
{
    if (root == NULL) // if tree is empty
        return pointer to node with key
    else
    {
        BSTnode a = getMode(key);
        if (a->second == value)
            return a;
    }
}

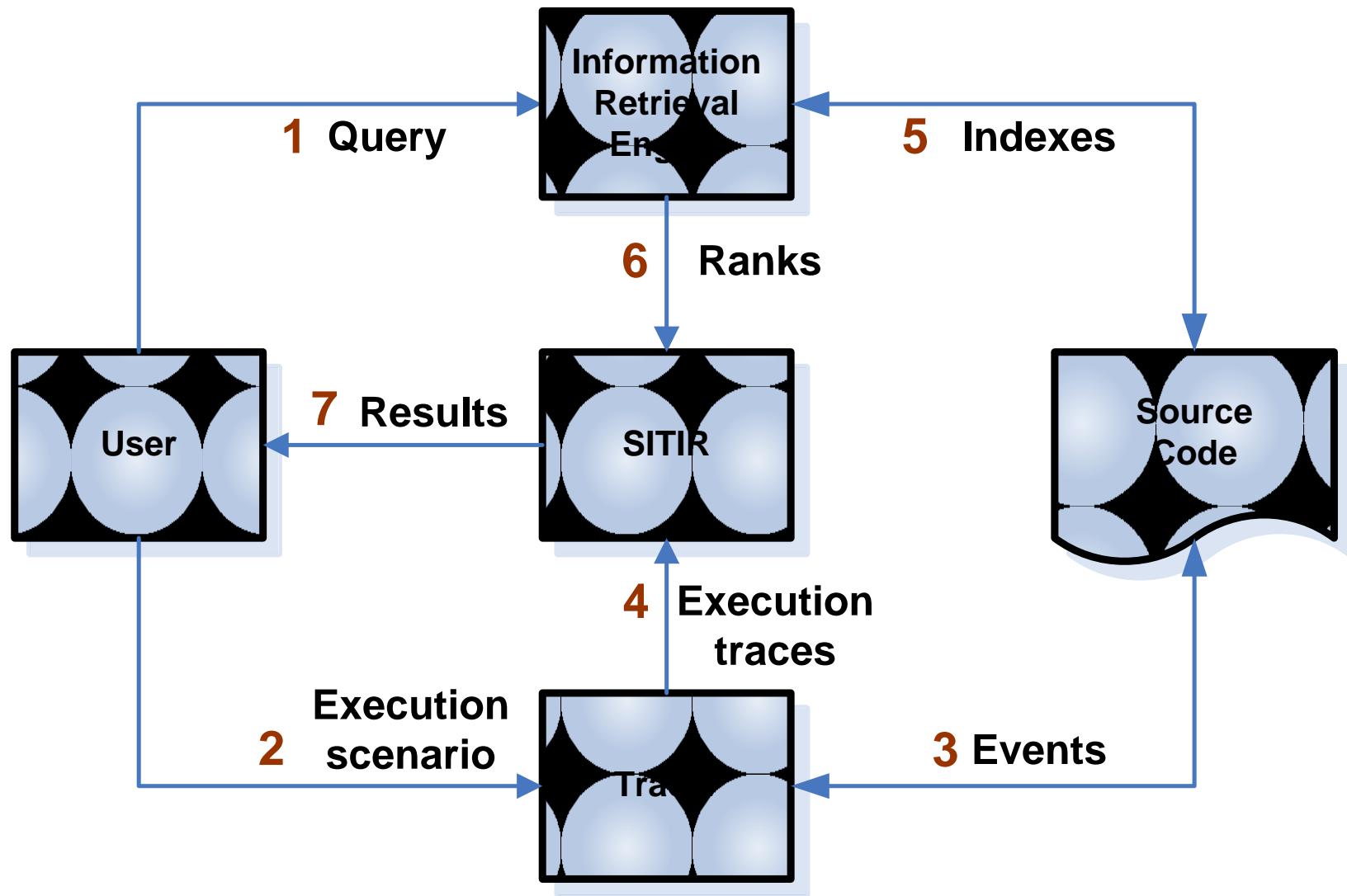
BSTnode BST_getMode(KeyType key)
{
    if (root == NULL) // if tree is empty
        return pointer to node with key
    else
    {
        pair<KeyType,ValueType> item(key, 0); // create item to be inserted
        if (_root == 0)
        {
            BSTnode new_node = new BSTnode(0, 0, item);
            root = new_node;
            return new_node;
        }
        pair<KeyType,ValueType> result(item, 0); // search for key
        if (!result.second) // location found
            return *result.first; // tree node exists
        else
        {
            BSTnode parent = result.first;
            BSTnode new_node = new BSTnode(0, 0, parent->item);
            if (parent->left == new_node)
                parent->left = new_node;
            else
                parent->right = new_node;
            return *new_node;
        }
    }
}

pair<KeyType,ValueType> BST_find(KeyType key)
{
    if (root == NULL) // if tree is empty
        return pair<KeyType,ValueType>(key, 0);
    else
    {
        pair<KeyType,ValueType> item(key, 0); // create item to be inserted
        if (_root == 0)
        {
            BSTnode new_node = new BSTnode(0, 0, item);
            root = new_node;
            return pair<KeyType,ValueType>(key, 0);
        }
        pair<KeyType,ValueType> result(item, 0); // search for key
        if (!result.second) // location found
            return result; // tree node exists
        else
        {
            BSTnode parent = result.first;
            BSTnode new_node = new BSTnode(0, 0, parent->item);
            if (parent->left == new_node)
                parent->left = new_node;
            else
                parent->right = new_node;
            return result;
        }
    }
}
  
```

## Information Retrieval Engine

Class	Method	Similarity
Camera	setScreenParams	0.813055
ScaleObjectTool	mouseDragged	0.807305
Camera	setScreenParamsParalle	0.80328
Camera	setSize	0.788526
ViewerCanvas	setScale	0.723998
GLCanvasDrawer	prepareView3D	0.705305
ViewerCanvas	scaleChanged	0.700639
JitterModule	setZScale	0.690434
JitterModule	setXScale	0.690117
ImageModule	setXScale	0.690104

# Feature Location with SITIR/FLAT<sup>3</sup>



# Collecting Execution Traces in SITIR

- Java Platform Debugger Architecture (JPDA)<sup>1</sup>
  - Infrastructure to build end-user debugging applications for Java platform
- JPDA highlights:
  - Debugger works on a separate virtual machine
  - Minimal interference of a tracing tool with a subject program
  - Separate thread-based traces
  - Marked traces (start/stop recording)

---

<sup>1</sup><http://java.sun.com/javase/technologies/core/toolsapis/jpda/>

# Indexing Software with Apache Lucene

- Parsing source code and extracting documents
  - corpus is a collection of documents (e.g., methods)
- Removing non-literals and stop words
  - common words in English, programming language keywords
- Preprocessing: split\_identifiers and SplitIdentifiers
- Indexing and retrieving semantic information with Lucene

# Parsing Source Code and Extracting Documents

- Corpus is a collection of documents (e.g., methods, classes, files)

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
)

public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
)

public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
)
```

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
)

public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
)

public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
)
```

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
)

public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
)

public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
)
```

# Parsing Source Code and Extracting Documents

- Corpus is a collection of documents (e.g., methods, classes, files)

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

if (monitor.isCanceled())
    throw new InterruptedException("The long running
    )
```

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

if (monitor.isCanceled())
    throw new InterruptedException("The long running
    )
```

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

if (monitor.isCanceled())
    throw new InterruptedException("The long running
    )
```

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

if (monitor.isCanceled())
    throw new InterruptedException("The long running
    )
```

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

if (monitor.isCanceled())
    throw new InterruptedException("The long running
    )
```

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

if (monitor.isCanceled())
    throw new InterruptedException("The long running
    )
```

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

if (monitor.isCanceled())
    throw new InterruptedException("The long running
    )
```

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

if (monitor.isCanceled())
    throw new InterruptedException("The long running
    )
```

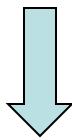
```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

if (monitor.isCanceled())
    throw new InterruptedException("The long running
    )
```

# Source Code is Text Too

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
    if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
    else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
    else
        processQueryString(monitor);

    if (monitor.isCanceled())
        throw new InterruptedException("The long running
    )
```



public void run IProgressMonitor monitor throws  
InvocationTargetException InterruptedException if m\_iFlag  
processCorpus monitor checkUpdate else if m\_iFlag  
processCorpus monitor UD\_UPDATECORPUS else  
processQueryString monitor if monitor isCancelled throw  
new InterruptedException the long running

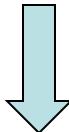
# Removing Stop Words

- Common words in English, programming language keywords

```
public void run IProgressMonitor monitor throws  
InvocationTargetException InterruptedException if  
m_iFlag the processCorpus monitor checkUpdate else  
if m_iFlag processCorpus monitor UD_UPDATECORPUS  
else a processQueryString monitor if monitor  
isCancelled throw new InterruptedException the  
long running
```

# Splitting Identifiers

```
public void run IProgressMonitor monitor throws InvocationTargetException  
InterruptedException if m_iFlag the processCorpus monitor checkUpdate else if  
m_iFlag processCorpus monitor UD_UPDATECORPUS else a processQueryString  
monitor if monitor isCancelled throw new InterruptedException the long  
running
```



- IProgressMonitor = i progress monitor
- InvocationTargetException = invocation target exception
- m\_IFlag = m i flag
- UD\_UPDATECORPUS = ud updatecorpus

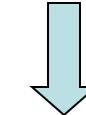
# Indexing Source Code with Lucene

```
public void run(IProgressMonitor monitor)
    throws InvocationTargetException,
           InterruptedException{
if ( m_iFlag == 0 )
    processCorpus(monitor,checkUpdate());
else if ( m_iFlag == 2 )
    processCorpus(monitor,UD_UPDATECORPUS);
else
    processQueryString(monitor);

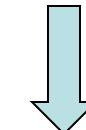
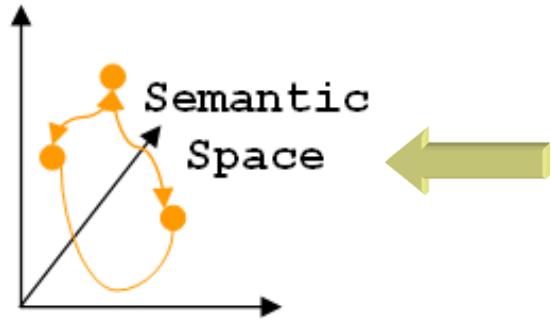
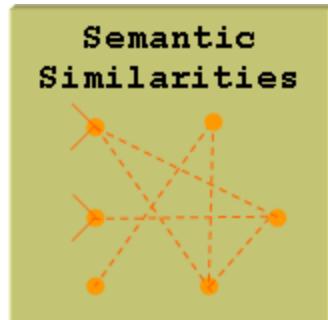
if (monitor.isCanceled())
    throw new InterruptedException("The long running
)
```



	process	flag	monitor
run	3	2	6
method1	x	x	x
method2	x	x	x
...	x	x	x



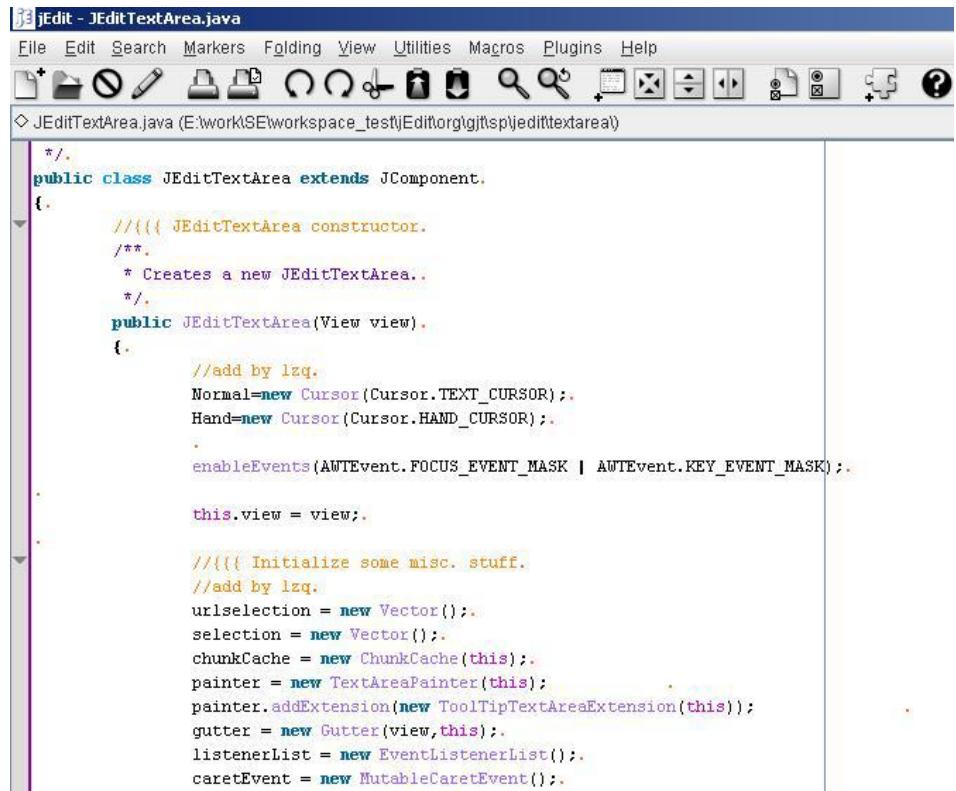
TF-IDF weighting



Similarity Measure:  
Cosine of the contained angle  
between the vectors

# Example of using SITIR

- Locating a feature in JEdit
- Feature: “showing white-space as a visible symbol in the text area”
- Steps:
  - Run a scenario
  - Run query
  - Explore results



The screenshot shows the JEdit text editor interface. The title bar reads "jEdit - JEditTextArea.java". The menu bar includes File, Edit, Search, Markers, Folding, View, Utilities, Macros, Plugins, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Replace. The main window displays the Java source code for JEditTextArea.java. The code defines a class that extends JComponent and implements several interfaces. It includes comments explaining the constructor and various methods like enableEvents and initialize misc. stuff.

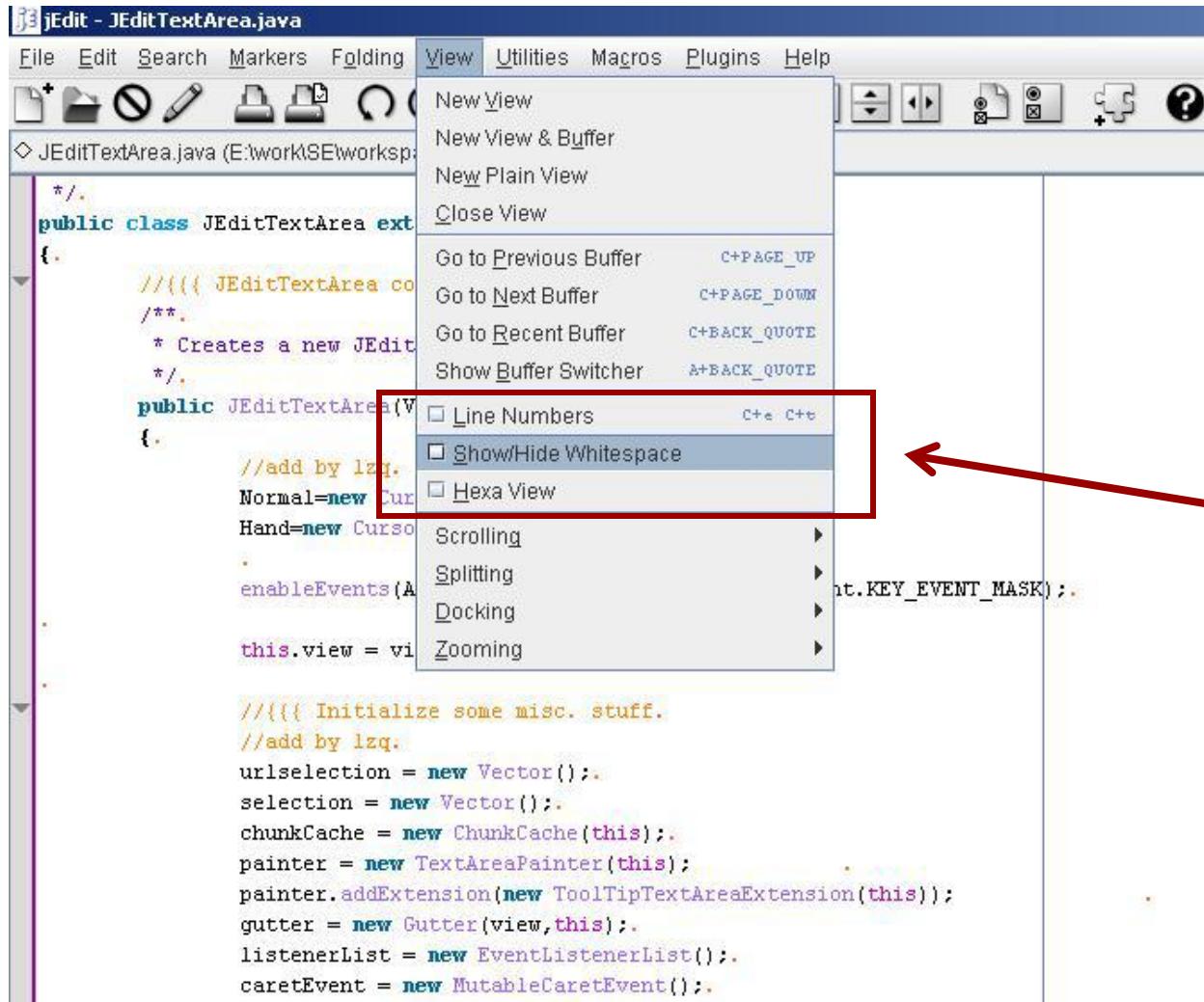
```
/*
 * JEditTextArea.java
 */
public class JEditTextArea extends JComponent {
    /**
     * JEditTextArea constructor.
     */
    * Creates a new JEditTextArea..
    */
    public JEditTextArea(View view) {
        //add by lzq.
        Normal=new Cursor(Cursor.TEXT_CURSOR);
        Hand=new Cursor(Cursor.HAND_CURSOR);

        enableEvents(AWTEvent.FOCUS_EVENT_MASK | AWTEvent.KEY_EVENT_MASK);

        this.view = view;

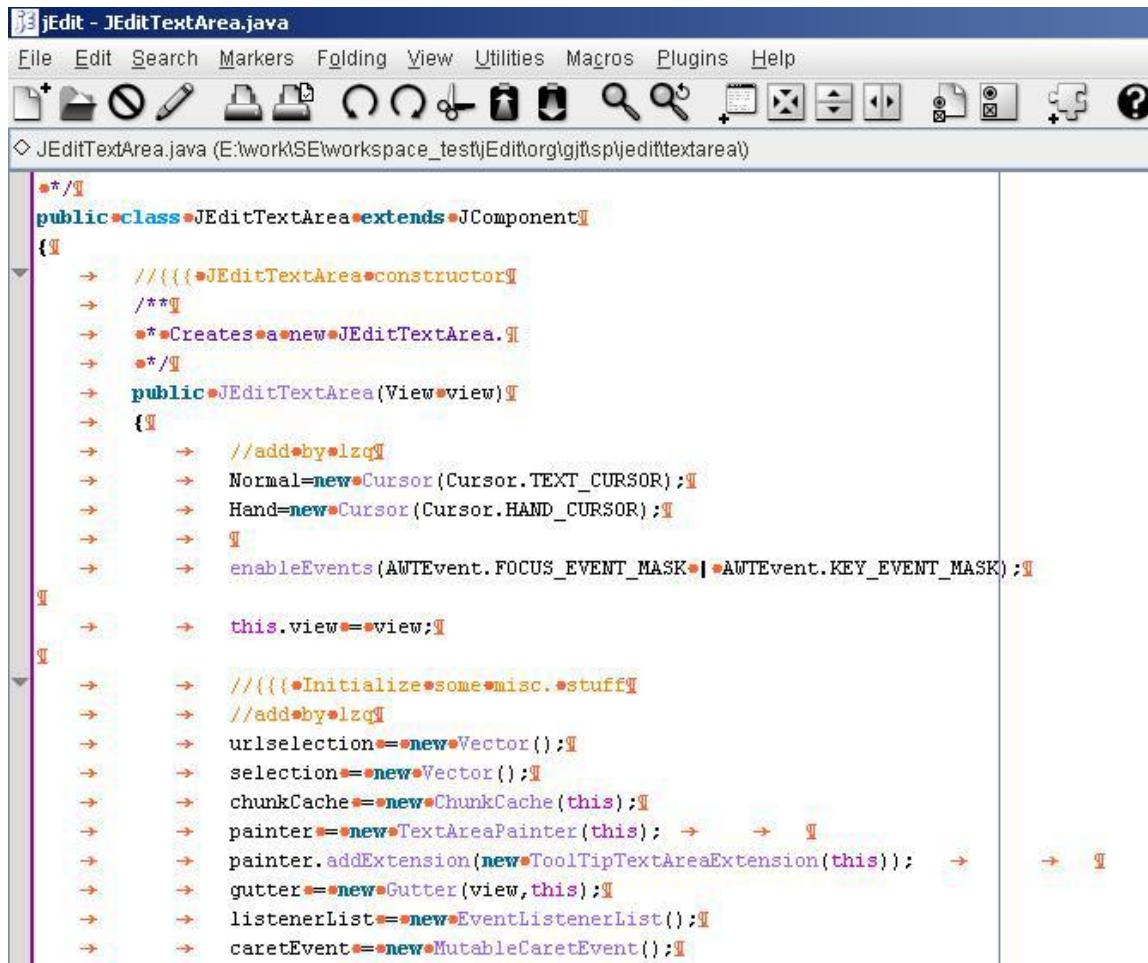
        //{{{{ Initialize some misc. stuff.
        //add by lzq.
        urlselection = new Vector();
        selection = new Vector();
        chunkCache = new ChunkCache(this);
        painter = new TextAreaPainter(this);
        painter.addExtension(new ToolTipTextAreaExtension(this));
        gutter = new Gutter(view,this);
        listenerList = new EventListenerList();
        caretEvent = new MutableCaretEvent();
    }
}
```

# Scenario Exercising the Feature



Start  
Tracing

# Scenario Exercising the Feature

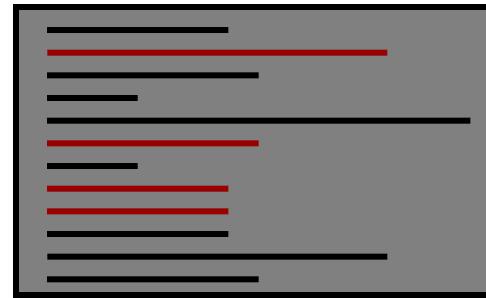
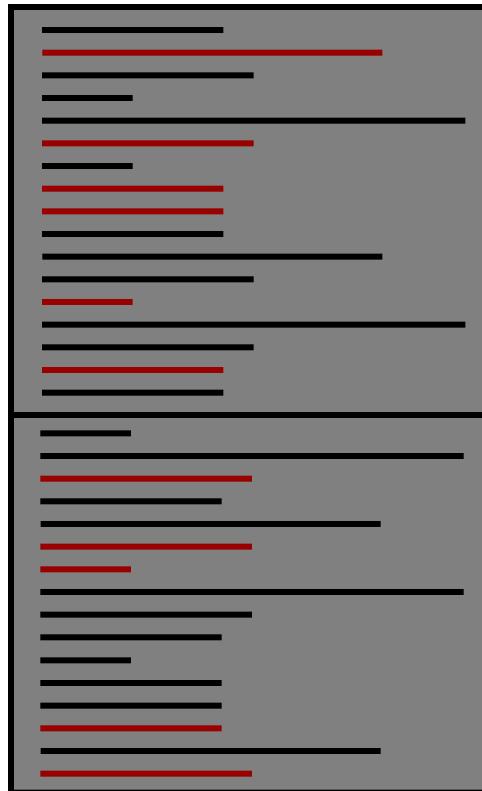


The screenshot shows the jEdit IDE interface with the title bar "jEdit - JEditTextArea.java". The menu bar includes File, Edit, Search, Markers, Folding, View, Utilities, Macros, Plugins, and Help. The toolbar contains various icons for file operations like Open, Save, Find, and Print. The central editor window displays the Java code for JEditTextArea.java:

```
/*
 * JEditTextArea.java
 */
public class JEditTextArea extends JComponent {
    // {{{ JEditTextArea constructor
    /**
     * Creates a new JEditTextArea.
     */
    public JEditTextArea(View view) {
        // add by slzq
        Normal=new Cursor(Cursor.TEXT_CURSOR);
        Hand=new Cursor(Cursor.HAND_CURSOR);
        enableEvents(AWTEvent.FOCUS_EVENT_MASK | AWTEvent.KEY_EVENT_MASK);
        this.view=view;
        // Initialize some misc. stuff
        // add by slzq
        urlselection=new Vector();
        selection=new Vector();
        chunkCache=new ChunkCache(this);
        painter=new TextAreaPainter(this);
        painter.addExtension(new ToolTipTextAreaExtension(this));
        gutter=new Gutter(view,this);
        listenerList=new EventListenerList();
        caretEvent=new MutableCaretEvent();
    }
}
```

Stop  
Tracing

# Example of using SITIR/FLAT<sup>3</sup> - Results



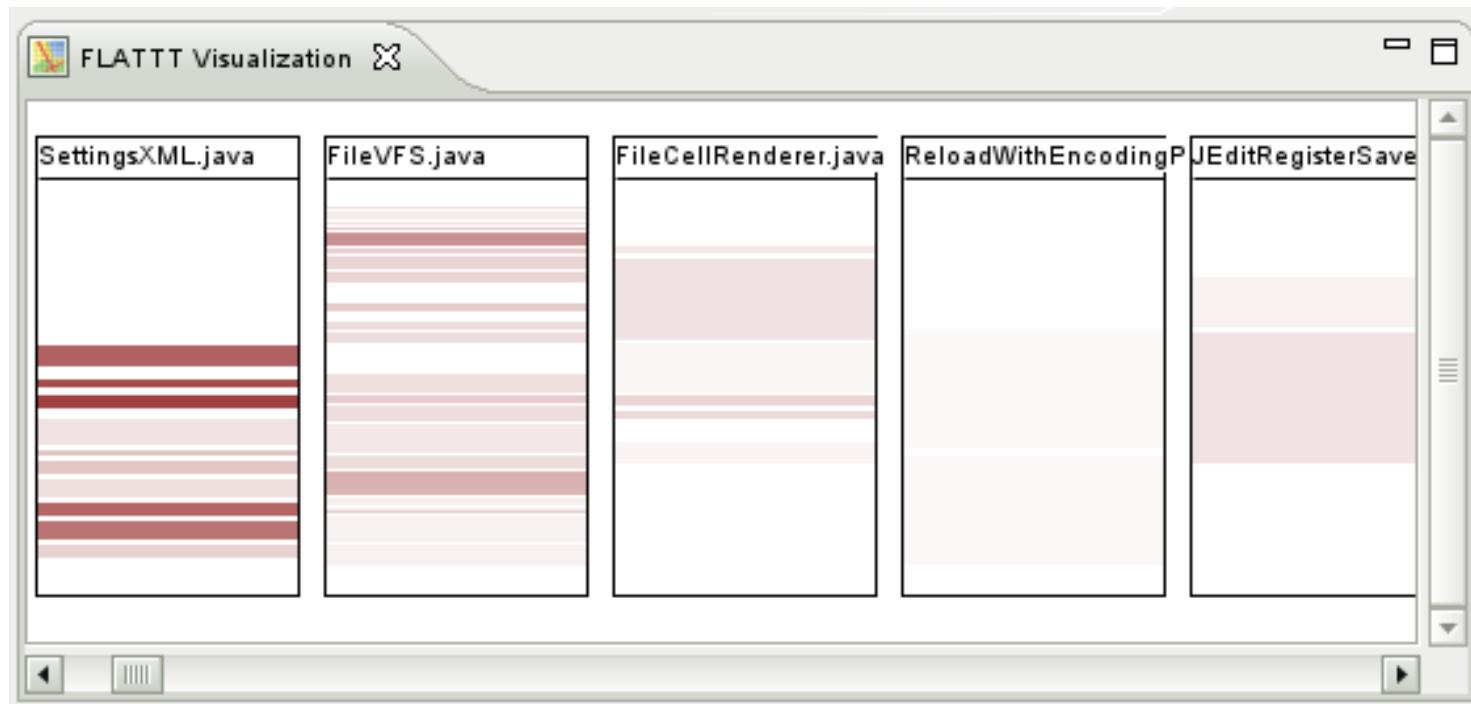
- Number of methods identified in the trace - **284**
- The position of the first relevant method according to IR ranking - **56**
- Position of the first relevant method according to SITIR - **7**

# DEMO: Locating Features in JEdit using FLAT<sup>3</sup>

- JEdit
  - 105KLOC
  - 910 classes
  - 5,530 methods
- Feature: “Word Count”
- Feature: “Save as”

# Other features

- Annotating features
- Visualizing results
- Saving/loading complex traces



# Acknowledgements

- ConcernMapper
- ConcernTagger
- MuTT
- AspectBrowser

# **FLAT<sup>3</sup>**

- Download the tool and complete source code at SEMERU web-site:
- <http://www.cs.wm.edu/semeru/flat3>



***SEMERU***