Research in Coding and IDEs

Jan-Peter Krämer
Media Computing Group
RWTH Aachen University

http://hci.rwth-aachen.de/cthci
Status Quo
Time in Software Development

[LaToza2006, Maintaining mental models: a study of developer work habits]
Software is complex and hard to understand.
Task context

- What is relevant information?
- What strategies are applied to find information?
Models for Developer Strategies

[Ko2006, An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks]

31 Professional Java Developers

5 Maintenance tasks
(3 Bugs, 2 Enhancements)

500 SLOC Java Paint Application
Models for Developer Strategies

[Ko2006, An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks]
Models for Developer Strategies

[Sillito2008, Asking and Answering Questions during a Programming Change Task]

9 experienced developers (pair programming)

1 of 5 maintenance tasks per session

ArgoUML 60k SLOC

16 developers from industry

Real world change task

Real world sour code
Models for Developer Strategies

[Sillito2008, Asking and Answering Questions during a Programming Change Task]

Finding focus points

Expanding focus points

Understanding a subgraph

Questions over groups of subgraphs
```java
package org.jhotdraw.contrib;

import org.jhotdraw.framework.DrawingView;

/**
 * The DesktopEvent event is very similar to an EventObject.
 * It gets passed to the EventObject when a DrawingView is added or
 * removed from the Desktop.
 * @author C.L.Gilbert <cmoy@users.sourceforge.net>
 * @version CURRENT_VERSION
 */

class DesktopEvent extends EventObject {
    private DrawingView myDrawingView;
    
    /**
     * Some events require the previous DrawingView (e.g. when a new DrawingView
     * is selected).
     */
    private DrawingView myPreviousDrawingView;

    public DesktopEvent(Desktop newSource, DrawingView newDrawingView) {
        this(newSource, newDrawingView, null);
    }

    public DesktopEvent(Desktop newSource, DrawingView newDrawingView, DrawingView newPreviousDV) {
        super(newSource);
        setDrawingView(newDrawingView);
        setPreviousDrawingView(newPreviousDV);
    }

    private void setDrawingView(DrawingView newDrawingView) {
        myDrawingView = newDrawingView;
    }

    public DrawingView getDrawingView() {
        return myDrawingView;
    }

    private void setPreviousDrawingView(DrawingView newPreviousDV) {
        myPreviousDrawingView = newPreviousDV;
    }

    public DrawingView getPreviousDrawingView() {
        return myPreviousDrawingView;
    }
}
```
Tools Used in Eclipse

[Murphy2006, How Are Java Software Developers Using the Eclipse IDE?]

![Pie chart showing navigation views used by Eclipse developers]

- **Package Explorer**: 74%
- **Search**: 11%
- **Type Hierarchy**: 10%
- **Outline**: 3%
- **Call Hierarchy**: 2%

Table 3: Top 10 commands executed across all 41 developers

<table>
<thead>
<tr>
<th>Command Identifier</th>
<th>Use (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.eclipse.ui.edit.delete</td>
<td>14.3</td>
</tr>
<tr>
<td>org.eclipse.ui.file.save</td>
<td>11.3</td>
</tr>
<tr>
<td>org.eclipse.ui.edit.text.goto.wordNext</td>
<td>7.3</td>
</tr>
<tr>
<td>org.eclipse.ui.edit.paste</td>
<td>6.8</td>
</tr>
<tr>
<td>org.eclipse.ui.edit.text.contentAssist.proposals</td>
<td>6.7</td>
</tr>
<tr>
<td>org.eclipse.ui.edit.text.goto.wordPrevious</td>
<td>5.9</td>
</tr>
<tr>
<td>org.eclipse.ui.edit.copy</td>
<td>4.6</td>
</tr>
<tr>
<td>org.eclipse.ui.edit.text.select.wordPrevious</td>
<td>3.4</td>
</tr>
<tr>
<td>org.eclipse.debug.ui.debugview.toolbar.stepOver</td>
<td>3.2</td>
</tr>
</tbody>
</table>

To help developers mark points of interest, Eclipse provides a variety of tools and views that are essential for efficient software development. These tools are designed to enhance productivity and ease of use by providing comprehensive views of the codebase, facilitating navigation, and supporting a wide range of development tasks.
Easing Access to Task Context
[Kersten2006, Using Task Context to Improve Programmer Productivity]
Recommender Tools

[Singer 2005, NavTracks: supporting navigation in software maintenance]
[DeLine 2005, Easing program comprehension by sharing navigation data]
[Čubranić 2005, Hipikat: recommending pertinent software development artifacts]

- Calculate a Degree of Interest for source code elements based on:
  - reading history
  - editing history
  - history of other team members
  - information from version control systems
Changing the Presentation

Changing the Presentation

[Bragdon2010, Code bubbles: a working set-based interface for code understanding and maintenance]
Bragdon2010, Code bubbles: a working set-based interface for code understanding and maintenance
Bragdon2010, Code bubbles: a working set-based interface for code understanding and maintenance
[Bragdon2010, Code bubbles: a working set-based interface for code understanding and maintenance]
Canvas Interfaces in the Wild

[DeLine2012, Debugger Canvas: Industrial experience with the code bubbles paradigm]

A. Usability Testing during Development

Fourteen weeks before the first release, we used the Rapid Iterative Testing and Evaluation method to improve the usability of the implementation. Briefly, we asked 10 participants to use Debugger Canvas to complete three tasks in one-hour sessions. The goal of each session was to see where the user struggled with the user experience and to gather feedback. After each user session, we identified any remaining critical usability problems and fixed them before the next session. While this method introduces too much variability between users to take controlled measures, the method is an efficient way to improve the tool and reduce overall participant frustration.

The most important usability problem we fixed in this process was our design decision to create a new canvas automatically for each debugging session. Our RITE users consistently debugged in many, short sessions (often focused on a single method) and therefore found the resulting canvases to be “clutter.” We updated the design so that debugging sessions all take place in the same canvas, unless the user explicitly creates a new one.

B. Download and Usage Data

We measured number of downloads for the tool, as well as number of users per day and per month. In the adoption numbers we were mostly looking for trends. We expect a non-useful tool to have bad word of mouth, leading to downloads going down sharply after the initial launch, while conversely, a useful tool should have a long tail after the initial spike, leading to a significant number of downloads beyond the first 2-3 weeks. The download curve is shown in Figure 5.

Figure 5: Number of unique downloads per week, after the initial release on 13 June 2011.

Downloads trends show a strong spike the first week, as would be expected, then settles into a mostly flat pattern. Downloads from week 3 and out represent 45% of the total. Given our initial criteria, this represents a positive result. It seems like Debugger Canvas may have enough usefulness that a relatively steady stream of users get pointed our way, despite no marketing activities from us after the initial launch, up until week 32 when we announced the second release.

The Microsoft Customer Experience Improvement Program (CEIP) provides the ability for customers to upload product usage data with complete anonymity. To participate in this program, users opt in to share their data with Microsoft, meaning that such data represents a self-selected sample of all users. (The Visual Studio team estimates that roughly 15% of their customers participate.) This data is then collated into counts of users who performed this action per day and per month.

To receive data, a team must instrument the operations in its product. For Debugger Canvas, we instrumented the operation of stepping with the debugger inside a code bubble, as well as our menu commands. Table 1 shows the frequency of Debugger Canvas’s operations, relative to stepping inside a code bubble (our most frequent instrumented operation).

Table 1. Relative use of Debugger Canvas’s commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Use relative to stepping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step into bubble</td>
<td>1.0</td>
</tr>
<tr>
<td>Create New Canvas</td>
<td>0.11</td>
</tr>
<tr>
<td>Start Debugging Without Debugger Canvas</td>
<td>0.07</td>
</tr>
<tr>
<td>Show Video Tutorial</td>
<td>0.03</td>
</tr>
<tr>
<td>Start Debugging With Debugger Canvas</td>
<td>0.03</td>
</tr>
<tr>
<td>Save As XPS</td>
<td>0.01</td>
</tr>
<tr>
<td>Send Feedback</td>
<td>0.01</td>
</tr>
<tr>
<td>Send As XPS Attachment</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Figure 6. Users per day who step into a code bubble at least once, as a percentage of usage on the first day. (The gap is due to missing data.) Figure 6 shows the number of users per day (in the CEIP sample) who step into a code bubble, starting in week 10 after release. (The gap is due to a problem with data collection in December 2011.) The trend of users per day is mostly flat and then picks up after the second release in week 32. When seen in the context of the trickle of new downloads in Figure 5, the overall curve in Figure 6 suggests that many users dropped out after initial use, but a large fraction continue to use it steadily.
Utilizing the Call Graph
The enthusiasm of participants was clearly evident and all asked to be notified of the tool's availability. The speed and success results for task 1 are summarized in Figure 13. For task 1, the number of successful participants was significantly higher for the Whyline group (10 Whyline participants succeeding, compared to 0 in the control). The median time on task was also lower for the Whyline group (20 minutes, compared to 30 minutes for the control group).

For task 2, the number of successful participants was also significantly higher for the Whyline group (8 Whyline participants succeeding, compared to 2 in the control). The median time on task was also lower for the Whyline group (15 minutes, compared to 20 minutes for the control group).

Results for task 1:
- # successful: Whyline > Control
- Time (min): Whyline < Control

Results for task 2:
- # successful: Whyline > Control
- Time (min): Whyline < Control
In practice: Feasible paths most interesting
[LaToza2010, Developers ask reachability questions]
Utilizing Call Graph Information

[LaToza2010, Searching Across Paths]

Legend

- public / protected / private method
- method visited by developer
- method with callers that are not shown
- type with type name
- method call that is always executed
- method call that might execute
- mutually exclusive method calls
- method call in a loop
- recursive method call
- paths of calls with hidden methods
- data flow
- expression
- expression that matches search
Static Analysis in the Wild

[Clang Static Analyzer, http://clang-analyzer.llvm.org/]
Call Hierarchy
Stacksplorer

[Karrer2011, Stacksplorer: Call Graph Navigation Helps Increasing Code Maintenance Efficiency]
Blaze

[Krämer2012, Blaze: Supporting Two-phased Call Graph Navigation in Source Code]
Analyzing Navigation Behavior
<table>
<thead>
<tr>
<th>Find Change Location</th>
<th>Task Success</th>
<th>Task Completion Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Side Effects of Change</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Xcode</th>
<th>Call Hierarchy</th>
<th>Stacksplorerer</th>
<th>Blaze</th>
</tr>
</thead>
</table>

33 Developers

80,000 Lines of Code

[Krämer2013, How Tools in IDEs Shape Developers' Navigation Behavior]
Task Success

\[ p = 0.015 \]

- Xcode
- Call Hierarchy
- Stacksplorer
- Blaze

# of successful participants
Task Completion Time

$p = 0.022$

<table>
<thead>
<tr>
<th>Tool</th>
<th>Total Time Required in Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xcode</td>
<td>1900</td>
</tr>
<tr>
<td>Call Hierarchy</td>
<td>1425</td>
</tr>
<tr>
<td>Stacksplorerer</td>
<td>950</td>
</tr>
<tr>
<td>Blaze</td>
<td>475</td>
</tr>
</tbody>
</table>
Effectiveness

Xcode

Call Hierarchy

Stacksplorer

Blaze

Efficiency

Xcode

Call Hierarchy

Stacksplorer

Blaze

Why?

UI Differences

Navigation Behavior
[Fouse2011, ChronoViz: A system for supporting navigation of time-coded data]
Comparing Navigation Behavior
$I_1 = (p_{1,1}, ..., p_{640,480})$

$I_2 = (p_{1,1}, ..., p_{1024,768})$

1. Features
2. Transformations
[Piorkowski 2011, Modeling programmer navigation: A head-to-head empirical evaluation of predictive models]
A Predictor

[Piorkowski2011, Modeling programmer navigation: A head-to-head empirical evaluation of predictive models]

\[ H=(m_1, \ldots, m_i) \]  
Navigation History  
\[ H = (a, b, a, d) \]

\[ M_i \]  
All methods known to developer at time \( i \)  
\[ M_4 = \{a, b, d\} \]

\[ A_i: M_i - \{m_i\} \rightarrow \mathbb{R} \]  
Activation value for each method in \( M_i \)  
\[ A_4(a) = 3 \]
\[ A_4(b) = 2 \]

\[ R_i: M_i - \{m_i\} \rightarrow \mathbb{N} \]  
Rank-transformed version of \( A_i \)  
\[ R_4(a) = 1 \]
\[ R_4(b) = 2 \]

Result: \( N \) top-ranked methods
Prediction Accuracy

**Xcode**

**Stacksplorer**

- Recency
- Frequency
- Working Set
- Within File Distance
- Fwd Call Depth
- Undirected Call Depth
- Bug Report Similarity

Prediction Accuracy: 41%
**Prediction Accuracy**

<table>
<thead>
<tr>
<th>Xcode</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stacksplorer</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Xcode**
  - N = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
  - Fwd Call Depth

- **Stacksplorer**
  - N = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
  - Undirected Call Depth

---

*CTHCl — Jan-Peter Krämer*
Forward Call Depth

Undirected Call Depth
Forward Call Depth

Undirected Call Depth

N = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

p=0.002

p=0.001

p=0.003

p=0.001
Outlook

Design

Analyze

Implement
<?xml version="1.0" encoding="utf-8"?>
  <mx:Script>
    <![CDATA[
      public function loadData():void {
        URLLoader
      }
    ]]>"
  </mx:Script>
</mx:Application>
// Introducing Codelets...
```javascript
// Tree

function drawTree() {
  var blossomPoints = [];

  resetRandom();
  drawBranches(0, -Math.PI/2, canvasWidth/2, canvasHeight, 30);
  resetRandom();
  drawBlossoms(blossomPoints);
}

function drawBranches(l, angle, x, y, width, blossomPoints) {
  ctx.save();
  var length = tween(l, 1, 60, 12, 3) + random(0.7, 1.3);
  if (l <= 0) { length = 97; }
  ctx.translate(x, y);
  ctx.rotate(angle);
  ctx.fillStyle = "rgba(0, 0, 0, 0.3)";
  ctx.fillRect(0, -width/2, length, width);
  ctx.restore();

  var tipX = x + (length - width/2) + Math.cos(angle);
  var tipY = y + (length - width/2) + Math.sin(angle);

  if (l >= 4) {
    blossomPoints.push([x, y, tipX, tipY]);
  }
  if (l <= 6) {
    drawBranches(l + 1, angle + random(-0.15, -0.05), Math.PI, blossomPoints);
    drawBranches(l + 1, angle + random(0.15, 0.05), Math.PI, blossomPoints);
  } else if (l < 12) {
    // Additional code for branch...
  }
}
```

[Victor2012, Inventing on Principle]
Summary

Finding focus points
Expanding focus points
Understanding a subgraph
Questions over groups of subgraphs