Houston: We are in overload

Gail Murphy
University of British Columbia

Note: This presentation involves many animations, videos, etc. Some of these are available on-line (avi files will attempt to play). Those that are not are indicated.

Material in this talk is licensed under a Creative Commons Attribution-Share Alike 3.0 Unported license.
unexpected
information overload
un·ex·pect·ed

coming without warning; unforeseen

bugs, “found design”, missing documentation, regressions, …
information overload

too much information to make a decision
public ZoomDemo() {

    // create a simple graph for the demo
    graph = TestGraphs.getOneComponentGraph();
    final Layout layout = new ISOMLayout(graph);
    PluggableRenderer pr = new PluggableRenderer();
    VisualizationViewer vv = new VisualizationViewer(layout, pr);

    // add my listener for ToolTips
    vv.setToolTipTextFunction(new DefaultToolTipFunction());
    // vv.setGraphMouse(new KGraphMouse(vv));
    vv.setPickSupport(new ShapePickSupport());

    // create a frame to hold the graph
    final JFrame frame = new JFrame();
    Container content = frame.getContentPane();
    JPanel panel = new JPanel(new BorderLayout());
    panel.add(vv);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    content.add(panel);
    frame.pack();
    frame.setVisible(true);
}

private void main(String[] args) {
    ZoomDemo zoomDemo = new ZoomDemo();
    zoomDemo.setVisible(true);
}
```
public ZoomDemo()
{
  // create a simple graph for the demo
  graph = TestGraphs.getOneComponentGraph();

  final Layout layout = new ISOMLayout(graph);
  PluggableRenderer pr = new PluggableRenderer();

  vv = new VisualizationViewer(layout, pr);

  // add my listener for ToolTips
  vv.setToolTipTextFunction(new DefaultToolTipFunction());
  vv.setGraphMouse(new KGGraphMouse(vv));

  vv.setPickSupport(new ShapePickSupport());

  // create a frame to hold the graph
  final JFrame frame = new JFrame();
  Container content = frame.getContentPane();
  JPanel panel = new JPanel(new BorderLayout());
  panel.add(vv);
  content.add(panel);
}
```
public VisualizationViewer(Layout layout, Renderer renderer, this(new DefaultVisualizationModel(layout), renderer);

/**
 * Create an instance with passed parameters.
 * @param layout The Layout to apply, with its
 * @param renderer The Renderer to draw it with
 * @param preferredSize the preferred size of this View
 */

public VisualizationViewer(Layout layout, Renderer renderer, this(new DefaultVisualizationModel(layout, preferredSize), renderer);

// protected boolean lockLayoutToViewSize;

protected Map locationMap = new HashMap();
/**
 * @param pickedState The pickedState to set.
 */

public void setPickedState(PickedState pickedState) {
    if (pickEventListener != null && this.pickedState != null) {
        this.pickedState.removeItemListener(pickEventListener);
    }
    this.pickedState = pickedState;
    if (renderer != null) {
        renderer.setPickedKey(pickedState);
    }
    if (pickEventListener == null) {
        pickEventListener = new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                repaint();
            }
        };
    }
    pickedState addItemListener(pickEventListener);
}

/**
 * @return Returns the pickSupport.
 */

public PickSupport getPickSupport() {
    return pickSupport;
}

/**
 * @param pickSupport The pickSupport to set.
 */

public void setPickSupport(PickSupport pickSupport) {
    this.pickSupport = pickSupport;
}
return;
pick(picked, false);
picked = null;
repaint();
}

public void mouseDragged(MouseEvent e) {
    if (picked == null)
        return;
    Point2D p = inverseViewTransform(e.getPoint());

    model.getGraphLayout().forceMove(picked, p.getX(), p.getY());
    repaint();
}

public void mouseMoved(MouseEvent e) {
    return;
}

/**
 * @see java.awt.event.MouseWheelListener#mouseWheelMoved(java.awt.event.MouseWheelEvent)
 */

public void mouseWheelMoved(MouseWheelEvent e) {
    return;
}
```java
return;
pick(picked, false);
picked = null;
repaint();

public void mouseDragged(MouseEvent e) {
    if (picked == null)
        return;
    Point2D p = inverseViewTransform(e.getPoint());
    model.getGraphLayout().forceMove(picked, p.getX(), p.getY());
    repaint();
}

public void mouseMoved(MouseEvent e) {
    return;
}

/**
 * @see java.awt.event.MouseWheelListener#mouseWheelMoved(java.awt.
 *      event.MouseWheelEvent)
 */
public void mouseWheelMoved(MouseWheelEvent e) {
    return;
}
```
information
source code
search results
file containment
type hierarchy
class structure
overload?

50 lines

3 search hits

60 file items

5 type items

25 class items
information overload?

source code grows to hundreds
search results and thousands
file containment for complete task
type hierarchy
tslass structure
mismatch

Cartoon not licensed for web version
this talk is about...

mismatches between programmers and tools

enabling focus and flow for programmers
understanding mismatches
understanding mismatches
mismatch #1

questions programmers ask
vs.
questions tools answer
Which type represents this domain concept?

Who implements this interface?

What are the differences between these types?

What are the parts of this type?

How are these types related?

What data can we access from this object?

What code is involved in the implementation of this behaviour?

Sillito, Murphy, De Volder, FSE 2006
What are the differences between these types?
public class KKLayout extends AbstractLayout {
    private double EPSILON = 0.1f;
    private int currentIteration;
    private int maxIterations = 2000;
    private String status = "KKLayout";

    private double L; // the ideal L
    private double k = 1; // arbitrary constant
    private double[][] dm; // distance matrix

    protected boolean adjustForGravity = true;
    protected boolean exchangeVertices = true;

    protected Vertex[] vertices;
    protected Coordinate[] xdata;
    protected Distance distance;
}

public class KKLayoutInt extends AbstractLayout {
    private static final Object EX_KEY = "EX"

    private int currentIteration;
    private int maxIterations = 2000;
    private String status = "KKLayoutInt";

    private int L; // the ideal length
    private static final double K = 10000;
    private int[] dm; // distance matrix

    protected boolean adjustForGravity = true;
    protected boolean exchangeVertices = true;

    protected Vertex[] vertices;
    protected Coordinate[] xdata;
}

*/
/**
 * Implements the Kamada-Kawai algorithm for network visualization.
 * Does not respect filter calls, and sometime it is very slow.
 *
 * @see "Tomihisa Kamada and Satoru Kawai: An algorithm for drawing general undirected graphs"
 * @see "Tomihisa Kamada: On visualization of network diagrams"
 *
 * @author Masanori Harada
 */

public class KKLayout extends AbstractLayout {

    private double EPSILON = 0.1d;

    private int currentIteration;
    private int maxIterations = 2000;
    private String status = "KKLayout";

    private double L; // the ideal length
    private double K = 1; // arbitrary constant
    private double[][] dm; // distance matrix

    private boolean adjustForGravity = true;
    private boolean exchangeVertices = true;

    private Vertex[] vertices;
    private Coordinates[] xydata;

    /**
     * Retrieves graph distances between vertex pairs.
     */
    protected Distance distance;
}
mismatch #2

information programmers need

vs.

information tools show
“In a delocalized plan, pieces of code that are conceptually related are physically located in non-contiguous parts of the program”

—— Soloway, Pinto, Letovsky, Littman and Lampert, CACM 1988

“the programmers’ central goal for each maintenance task was to collect a working set of task-relevant code fragments”

—— Ko, Aung, Myers, ICSE 2005
mismatch #3

strategies programmers use

vs.

strategies tools support
click to play animations (avi)

Allen, Murphy, de Alwis, 2007
<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 1</td>
<td>Data 2</td>
<td>Data 3</td>
<td>Data 4</td>
</tr>
<tr>
<td>Data 5</td>
<td>Data 6</td>
<td>Data 7</td>
<td>Data 8</td>
</tr>
<tr>
<td>Data 9</td>
<td>Data 10</td>
<td>Data 11</td>
<td>Data 12</td>
</tr>
</tbody>
</table>

_Shepherd, Murphy, 2007_
this talk is about...

mismatches between programmers and tools

1. questions
2. information enabling focus and flow for
3. strategies programmers
flow  focus
focus  flow
focus  focus
flow

click to play animations (avi)
enabling focus and flow

structure
mismatch #1: questions

currently
mismatch #1: questions

with Ferret

de Alwis, Murphy, 2007
mismatch #1: questions
with Ferret

de Alwis, Murphy, 2007
mismatch #1: questions

with Ferret

de Alwis, Murphy, 2007
mismatch #1: questions

with Ferret

focus

flow

based on a small diary study

de Alwis, Murphy, 2007
mismatch #2: information

Kersten, Murphy, 2007
mismatch #2: information

with Mylyn

Kersten, Murphy, 2007
mismatch #2: information

with Mylyn

click to play screencast (avi)
(demo portion unavailable on-line)

Kersten, Murphy, 2007
mismatch #2: information

with Mylyn

focus  flow

based on a field study
and huge user community (1M+ downloads)

Kersten, Murphy, 2007
mismatch #3: strategies

currently

click to play screencast (avi)
mismatch #3: strategies

with web-style navigation

[click to play demo (avi)]
mismatch #3: strategies

with web-style navigation

*demo unavailable on-line*

*Sherwood, Murphy, 2007*
mismatch #3: strategies

with web-style navigation

focus    flow

*tbd*
a lurking problem

video used in talk unavailable on-line
a looming problem
a looming problem
more artifacts, looser structure
what to do?

1. enhance system structure model

2. rethink the model(s) behind the tools
meghan allen
john anvik
elisa baniassad
wesley coelho
davor cubranic
brian de alwis
rob elves
thomas fritz
jan hannemann
lyndon hiew
reid holmes
mik kersten
seonah lee
shawn minto
martin robillard
izzet safer
david shepherd
ducky sherwood
annie ying
trevor young
robert walker
and others!
SO...

Information
Information
Information
Information
Information
Information
Information
Information

Cartoon not licensed for web version
{add, compute, present} information with focus and flow to manage information effectively