Navigating Code

CS 695 / SWE 699: Programming Tools

Fall 2023

Today

- Part 1 Lecture(~60 mins)
  - 10 min break
- Part 2: Tech Talks (30 mins)
  - Two tech talks
- Part 3: In-Class Activity(40 mins)

Logistics

- HW 4 checkpoint due today
- HW 4 due 11/29

Overview

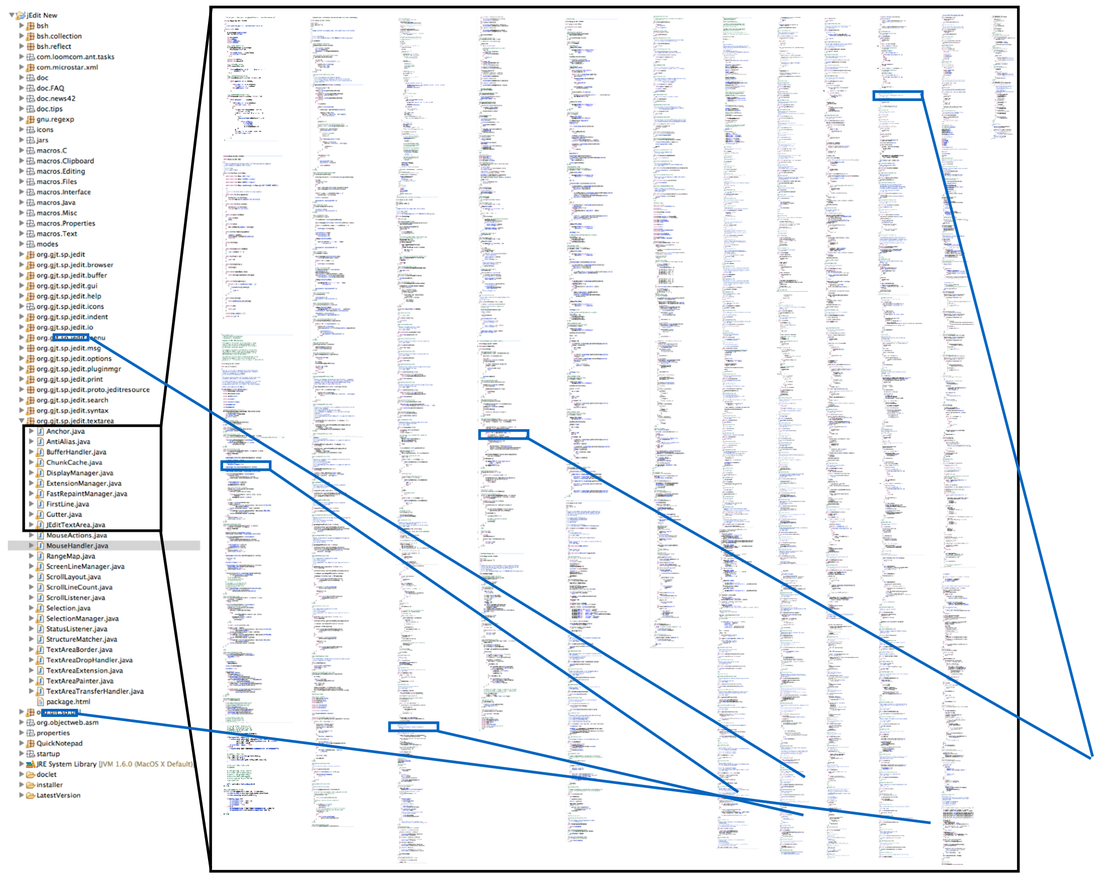
- What is code navigation?
- What are concerns?
- Code navigation tools

Code navigation: examples

- A developer wants to find method that implements *x.*
- A developer wants to find all of the methods involved in feature x.
- A developer wants to understand what a method does or when it is called.
- A developer wants to understand how to reuse a function by finding examples of code snippets.
- A developer wants to switch back to a method they were just editing.

Task context

- Could be
  - Set of **information necessary** to complete a task
  - Set of locations in **code** that must be **edited** to implement a change (e.g., add feature, fix bug)
- Which is it? Often used interchangeably…
- Sometimes known as a “working set”



How Effective Developers Investigate Source Code

- Unsuccessful subjects made all of their code modifications in one place even if they should have been **scattered** to better align with the existing design.
  - --> better support navigating across methods
- Program segments that were clearly relevant to the change task were not acknowledged when displayed **accidentally**.
  - --> suport intentional searches
- The successful subjects created a detailed and complete **plan** prior to the change whereas the unsuccessful and average subjects did not.
  - --> support building a change plan
- Successful subjects did not **reinvestigate** methods as frequently as unsuccessful subjects.
  - --> support understanding methods
- The successful subjects performed mostly **structurally** guided searches (e.g., keyword and cross-reference searches), rather than searches based on intuition (browsing) or aligned with the file decomposition of the system (scrolling).
  - --> support structural relationship traversal

Martin P. Robillard, Wesley Coelho, and Gail C. Murphy. 2004. How Effective Developers Investigate Source Code: An Exploratory Study. IEEE Trans. Softw. Eng. 30, 12 (December 2004), 889-903. DOI=http://dx.doi.org/10.1109/TSE.

Structural Relationship Traversal

- Developer is currently viewing an element in code
  - e.g,. class, method, statement, field reference
- Developers wishes to navigate to a **related** method
  - By reference, call, data dependency, …

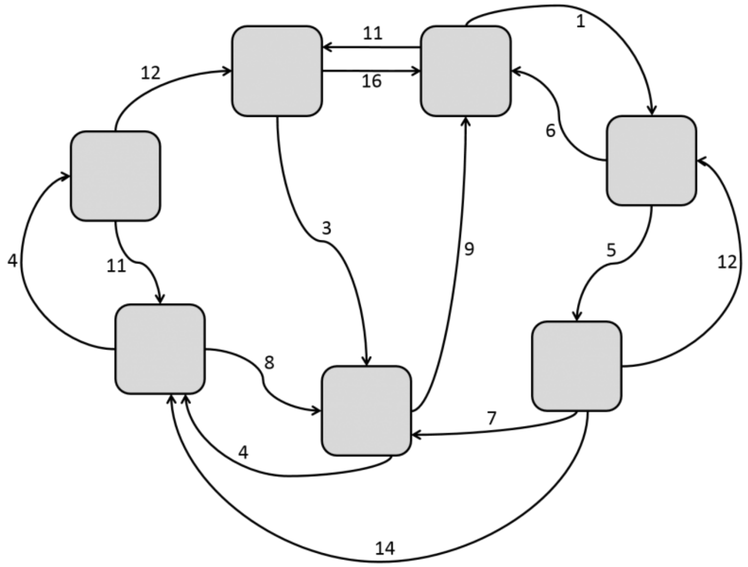
- How do developers make navigation decisions?

Information foraging

- Mathematical model describing navigation
- Analogy: animals foraging for food
  - Can forage in different patches (locations)
  - Goal is to maximize chances of finding **prey** while minimizing time spent in hunt
- Information foraging: navigating through an information space (patches) in order to maximize chances of finding prey (information) in minimal time

Information environment

- Information environment represented as **topology**
  - Information patches connected by traversable **links**
  - For SE, usually modeled as call graphs
    - methods are nodes and function invocations are edges



Traversing links

- Links - connection between patch offered by the information environment
- Cues - information features associated with outgoing links from patch
  - E.g., text label on a hyperlink
- User must choose which, of all possible links to traverse, has best chance of reaching prey

Scent

- User interprets cues on links by likelihood they will reach prey
  - e.g., do I think that the “invoke” method is likely to implement the functionality I’m looking for?

Simplified mathematical model

- Users make choices to maximize **possibility** of reaching prey per cost of interaction
- Predators (idealized) choice = max [V / C]
  - V - value of information gain, C - cost of interaction
- Don’t usually know ground truth, have to estimate
- Predator’s desired choice = max [E[V] / E[C]]

What’s a concern?

Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects. We know that a program must be correct and we can study it from that viewpoint only; we also know that it should be efficient and we can study its efficiency on another day, so to speak. In another mood we may ask ourselves whether, and if so: why, the program is desirable. **But nothing is gained —on the contrary!— by tackling these various aspects simultaneously. It is what I sometimes have called "the separation of concerns", which, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts, that I know of. This is what I mean by "focusing one's attention upon some aspect": it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect's point of view, the other is irrelevant.**

—Edsger W. Dijkstra. ”On the role of scientific thought”. 1974. EWD447.

Crosscutting concerns

- Ideal: one concern per module

- But, in practice modules exhibit
  - Scattering — single concern implemented in many modules
  - Tangling —- single module containing many concerns

Significant time spent navigating across task context

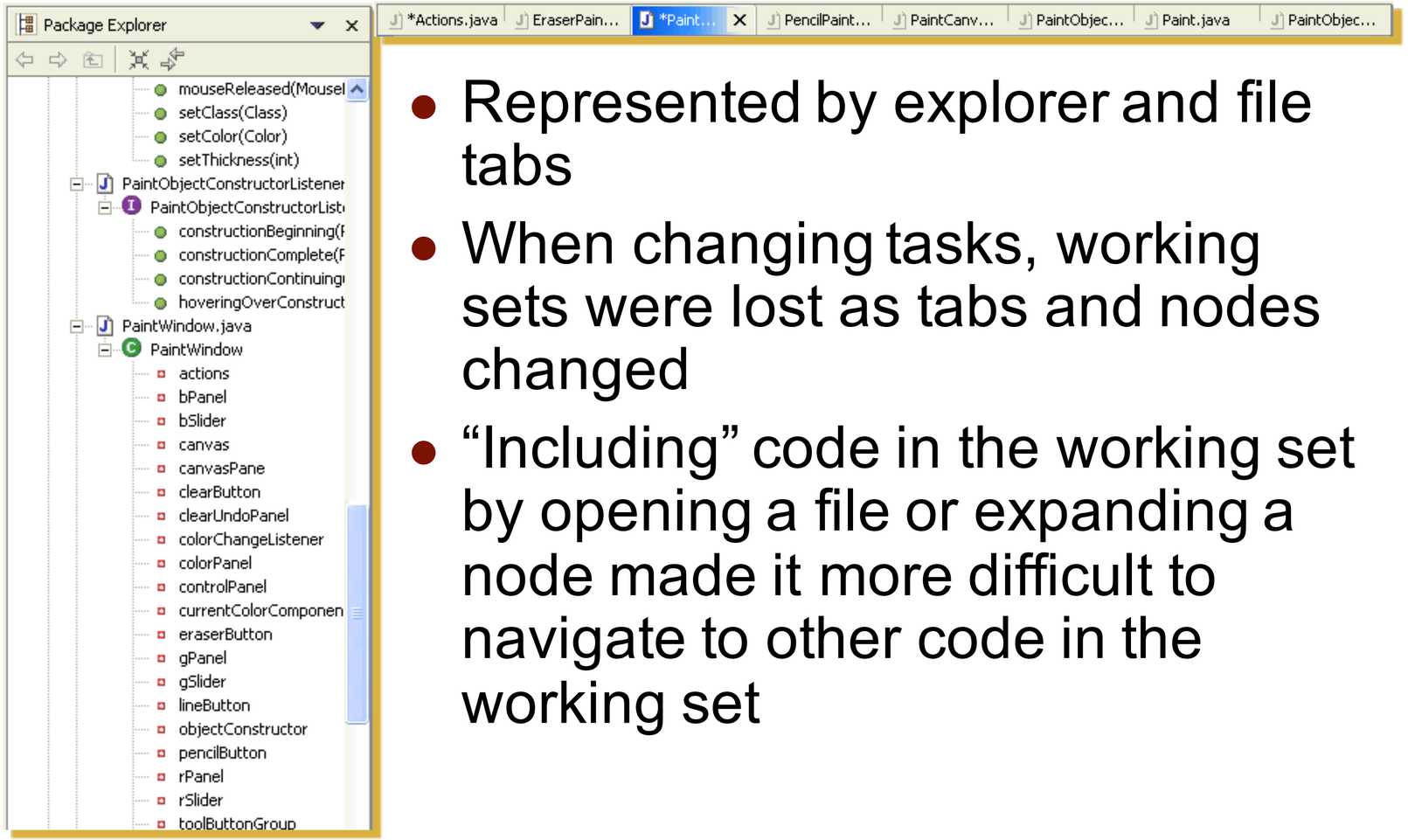
- Each instance of an interactive bottleneck cost only a few seconds, but . . .

Interactive Bottleneck	Overall Cost
Navigating to fragment in *same* file (*via scrolling*)	~ 11 minutes
Navigating to fragment in *different* file (*via tabs and explorer*)	~ 7 minutes
Recovering working set after returning to a task	~ 1 minute
Total Costs	~19 minutes

= 35% of uninterrupted work time!

Andrew J. Ko, Htet Aung, and Brad A. Myers. 2005. Eliciting design requirements for maintenance-oriented IDEs: a detailed study of corrective and perfective maintenance tasks. International conference on Software engineering,126-135.

Switching tasks incurs startup cost rebuilding task context



- Represented by explorer and file tabs
- When changing tasks, working sets were lost as tabs and nodes changed
- “Including” code in the working set by opening a file or expanding a node made it more difficult to navigate to other code in the working set

Andrew J. Ko, Htet Aung, and Brad A. Myers. 2005. Eliciting design requirements for maintenance-oriented IDEs: a detailed study of corrective and perfective maintenance tasks. International conference on Software engineering,126-135.

DeLine's study of developers

- Confirmed Ko's observation that:
  - Navigating and "re-finding" areas of the code that had already been visited was frequent, difficult and distracting
  - Textual searching and returning
  - Tabs got problematic when many opened
- All subjects wanted better inline comments and overview documentation.
- Wanted code annotations
- All subjects agreed that finding the entry point and understanding the control flow was the most difficult task

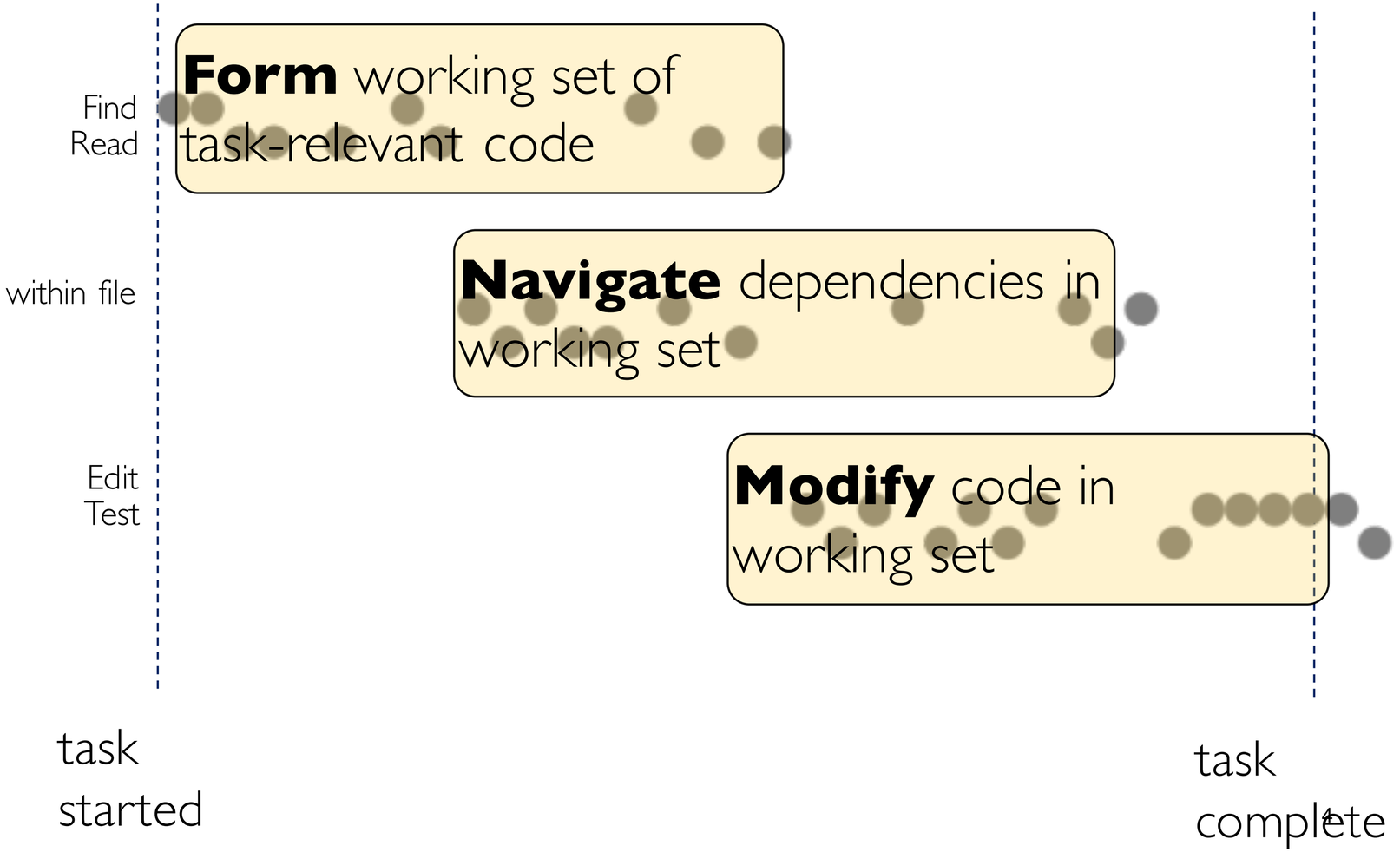
Robert DeLine, Amir Khella, Mary Czerwinski, and George Robertson. 2005. Towards understanding programs through wear-based filtering. Symposium on Software visualization (SoftVis '05), 183-192.

Field study of developers at IBM

- 8 IBM developers doing their own tasks using Eclipse for Java
- Interviews and 2-hour observations of actual use
- Experts do become disoriented
  - Did use Eclipse's advanced navigation tools, like find-all-callers
  - No trace of how got to the current file, or how to get back
  - Thrashing to view necessary context
- No support for switching tasks

Gail C. Murphy, Brian de Alwis, "Using Visual Momentum to Explain Disorientation in the Eclipse IDE", IEEE Symposium on Visual Languages and Human-Centric Computing, p. 51-54, , 2006

Working with task context



Andrew J. Ko, Htet Aung, and Brad A. Myers. 2005. Eliciting design requirements for maintenance-oriented IDEs: a detailed study of corrective and perfective maintena
conference on Software engineering,126-135.

Code navigation tools

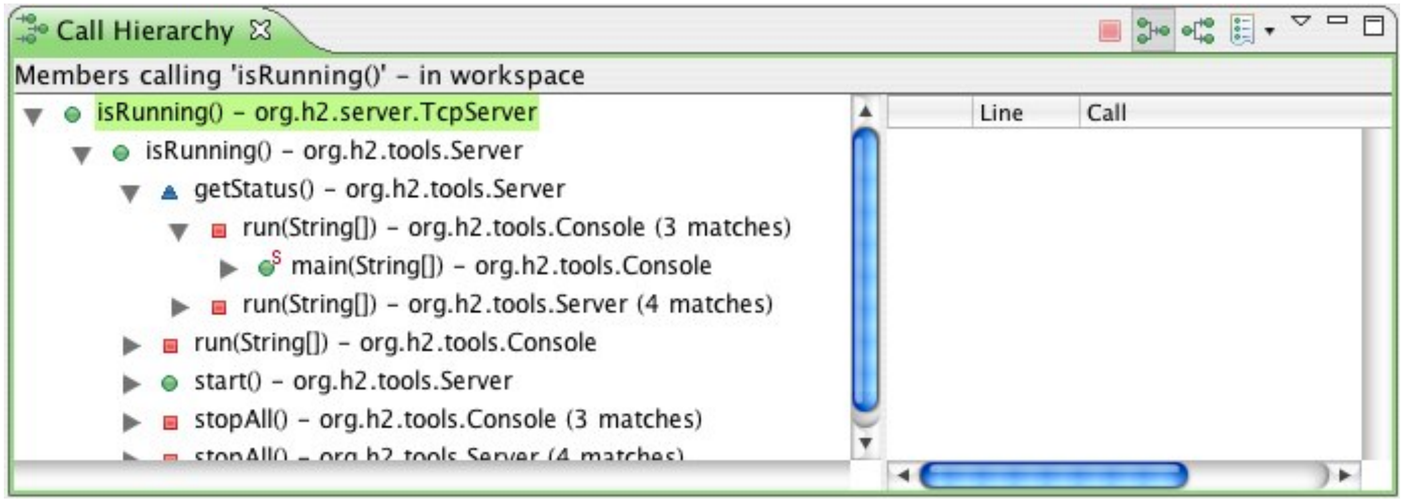
- Structural relationship traversal
  - Find starting point, traverse relationships to find other related code locations

- Recommenders
  - Based on {edits, navigation} past developers did on similar tasks, predict relevant elements

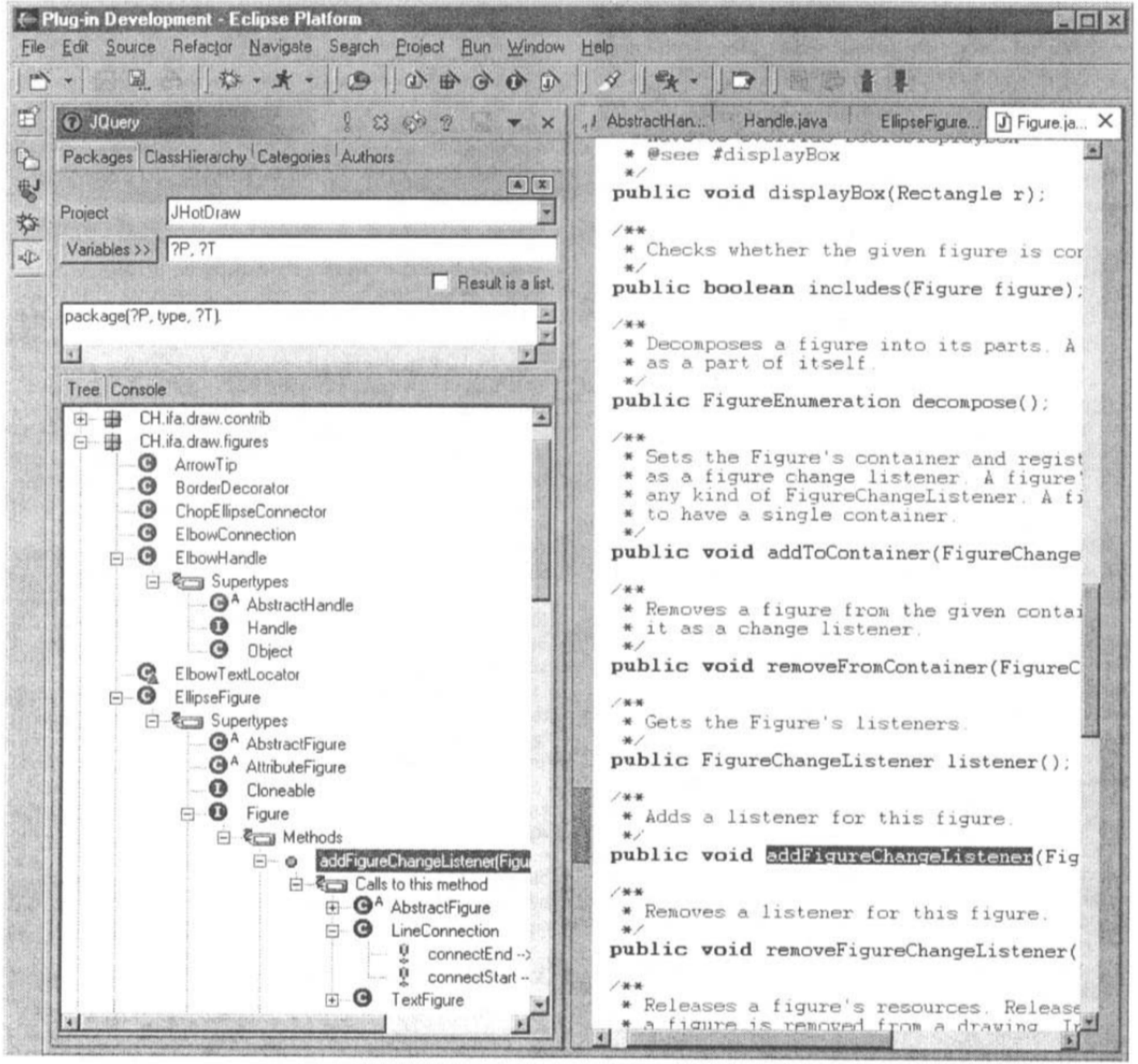
- Working set navigation
  - Make it easier to navigate back and forth between task context elements
  - Make it easier to resume tasks by redisplaying working set

Structural relationship traversal

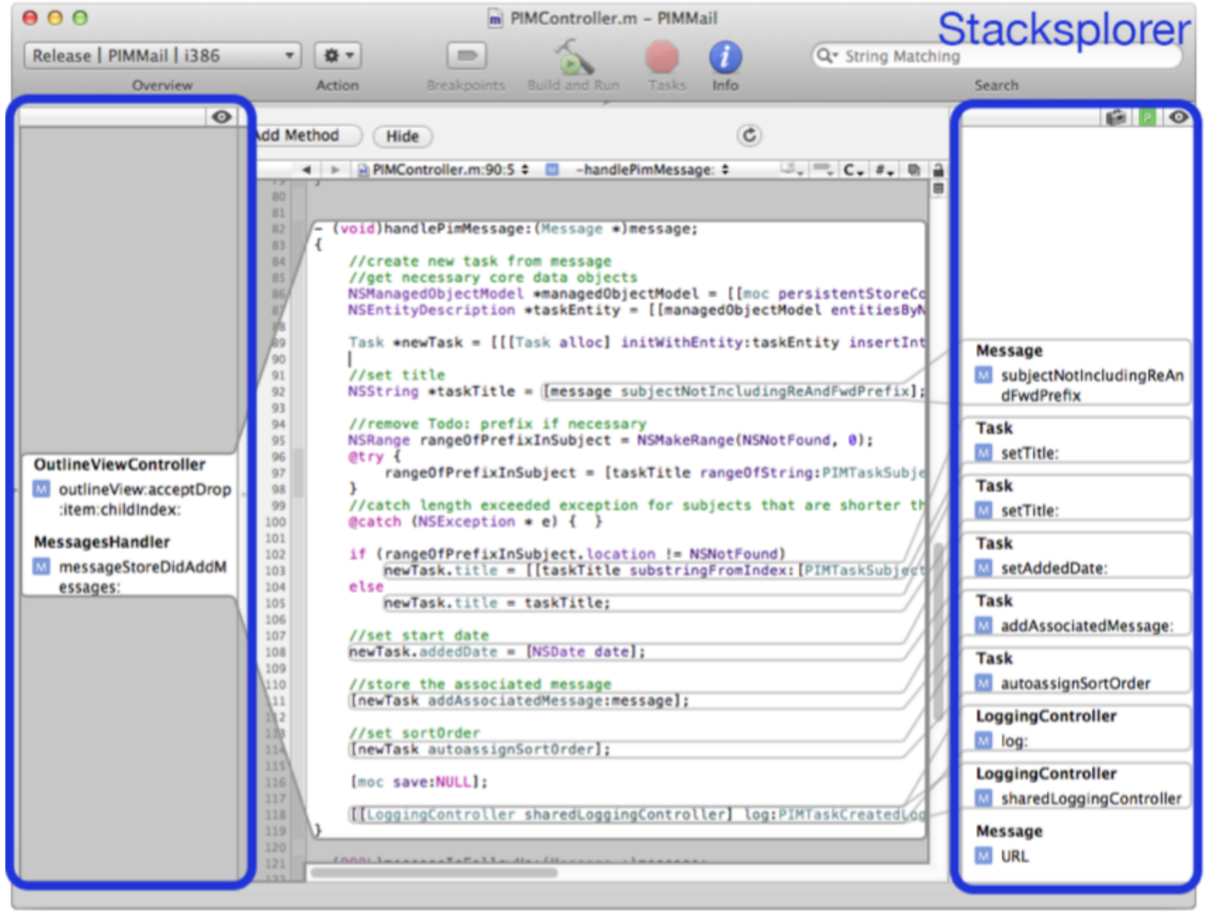
Call hierarchy view



JQuery


Plug-in Development - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
JQuery
Packages ClassHierarchy Categories Authors
Project JHotDraw
Variables >> ?P, ?T
Result is a list.
package(?P, type, ?T).
Tree Console
CH.ifa.draw.contrib
CH.ifa.draw.figures
ArrowTip
BorderDecorator
ChopEllipseConnector
ElbowConnection
ElbowHandle
Supertypes
AbstractHandle
Handle
Object
ElbowTextLocator
EllipseFigure
Supertypes
AbstractFigure
AttributeFigure
Cloneable
Figure
Methods
addFigureChangeListener(Figu
Calls to this method
AbstractFigure
LineConnection
connectEnd -->
connectStart --
TextFigure
AbstractHan...
Handle.java
EllipseFigure...
Figure.ja...
* @see #displayBox
*/
public void displayBox(Rectangle r);
/**
* Checks whether the given figure is cor
*/
public boolean includes(Figure figure);
/**
* Decomposes a figure into its parts. A
* as a part of itself.
*/
public FigureEnumeration decompose();
/**
* Sets the Figure's container and regist
* as a figure change listener. A figure'
* any kind of FigureChangeListener. A fi
* to have a single container.
*/
public void addToContainer(FigureChange
/**
* Removes a figure from the given contai
* it as a change listener.
*/
public void removeFromContainer(FigureC
/**
* Gets the Figure's listeners.
*/
public FigureChangeListener listener();
/**
* Adds a listener for this figure.
*/
public void addFigureChangeListener(Fig
/**
* Removes a listener for this figure.
*/
public void removeFigureChangeListener(
/**
* Releases a figure's resources. Release
* a figure is removed from a drawing. It

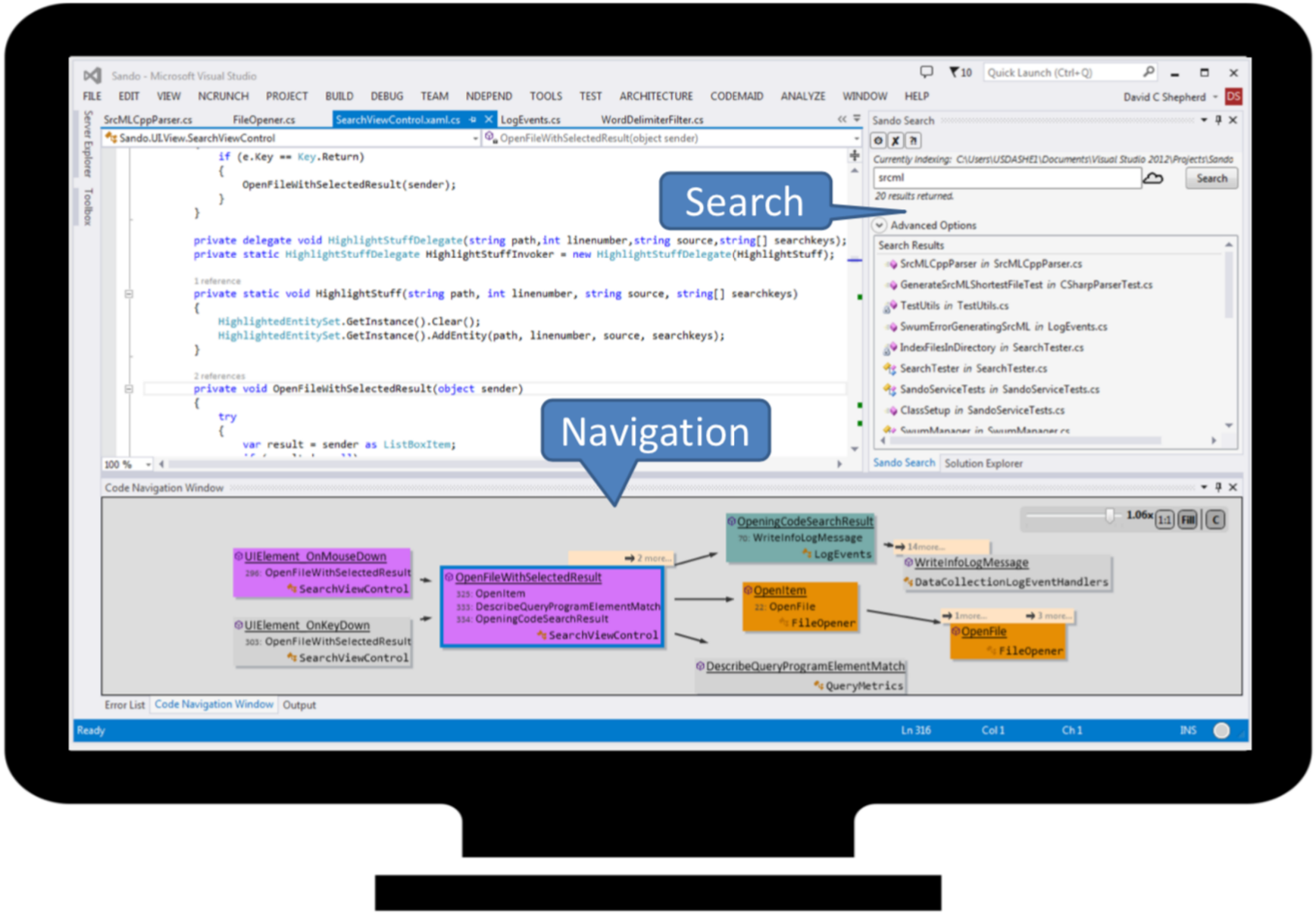
StackSplorer



https://dl.acm.org/doi/10.1145/2047196.2047225

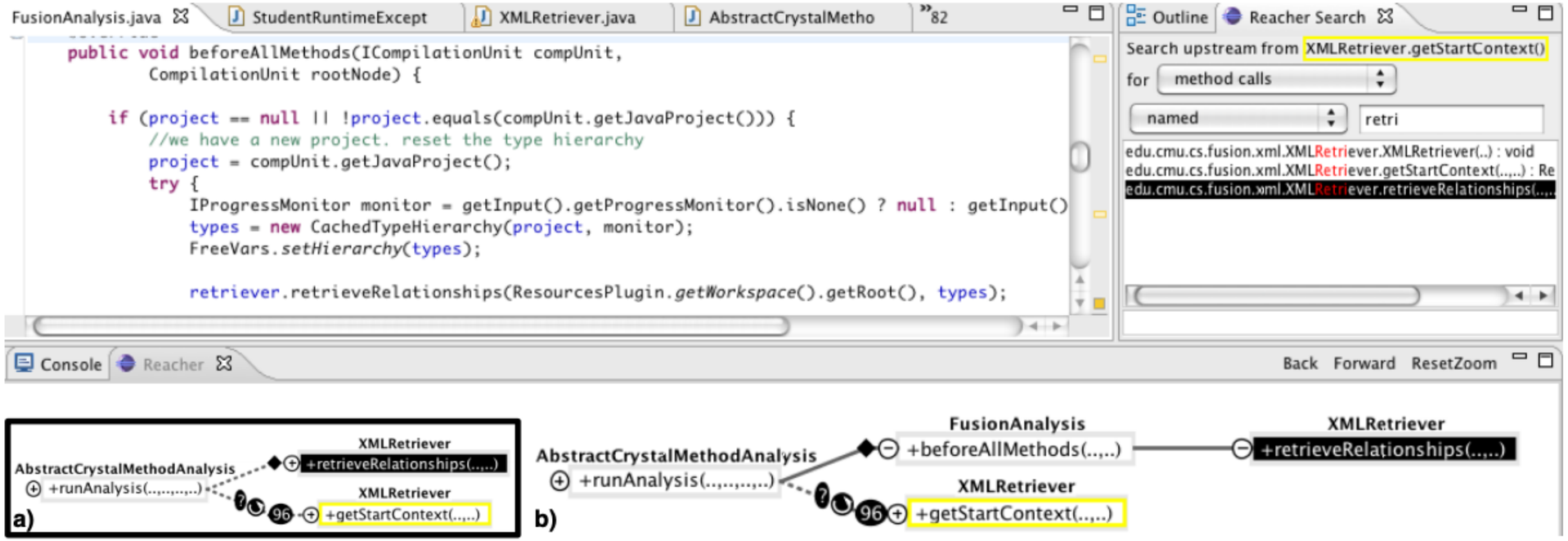
Thorsten Karrer, Jan-Peter Krämer, Jonathan Diehl, Björn Hartmann, and Jan Borchers. 2011. Stacksplorer: call graph navigation helps increasing code maintenance efficiency. In Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST '11). ACM, New York, NY, USA, 217-224. DOI: https://doi.org/10.1145/2047196.2047225

Prodet



Vinay Augustine, Patrick Francis, Xiao Qu, David Shepherd, Will Snipes, Christoph Bräunlich, and Thomas Fritz. 2015. A field study on fostering structural navigation with prodet. In Proceedings of the 37th International Conference on Software Engineering - Volume 2 (ICSE '15), Vol. 2. IEEE Press, Piscataway, NJ, USA, 229-238.

Reacher

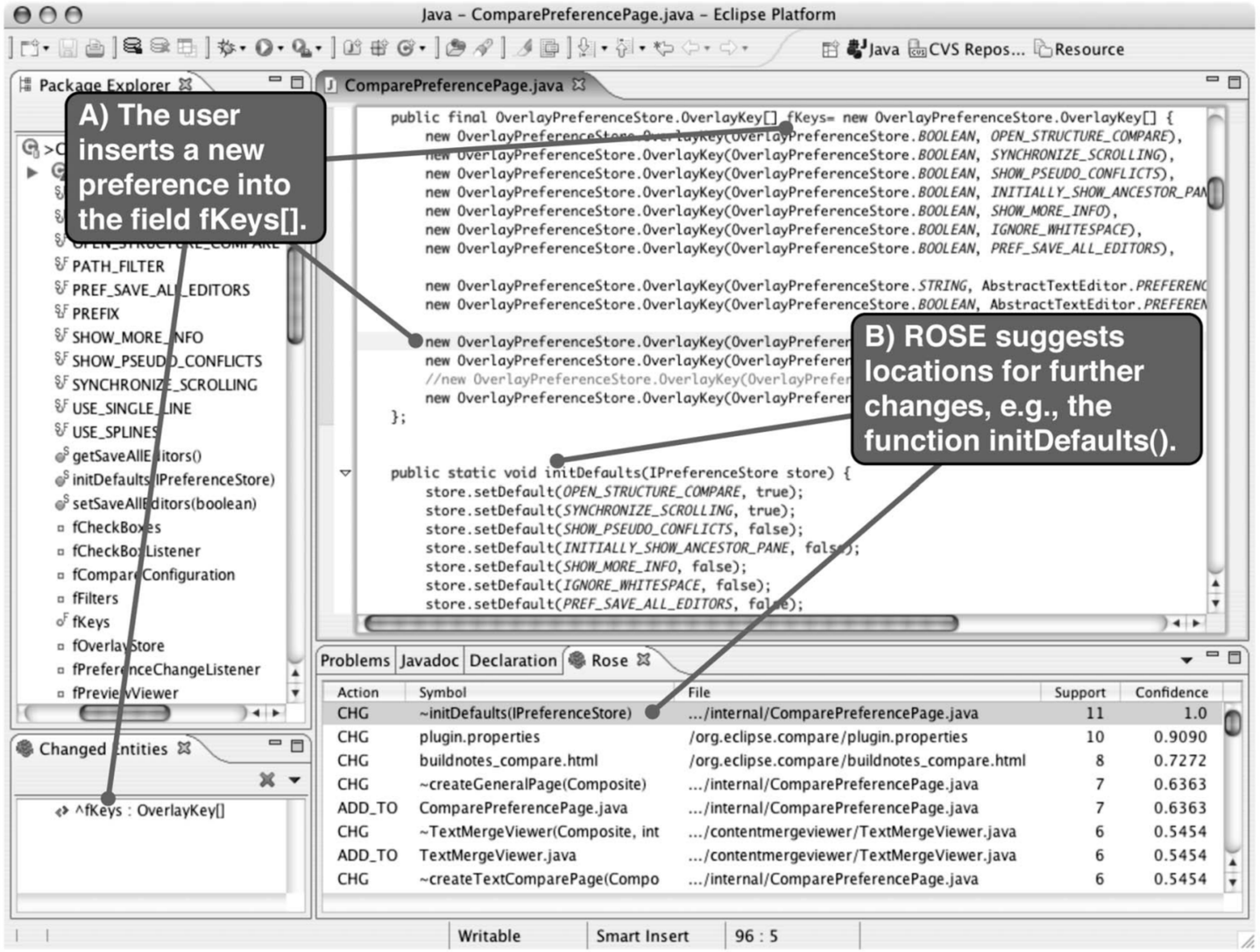


T. D. LaToza and B. A. Myers, "Visualizing call graphs," *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Pittsburgh, PA, 2011, pp. 117-124. doi: 10.1109/VLHCC.2011.6070388

Recommenders

- Based on {edits, navigation} past developers did on similar tasks, predict relevant elements

Rose



Thomas Zimmermann, Peter Weißgerber, Stephan Diehl, Andreas Zeller. Mining Version Histories to Guide Software Changes. IEEE Trans. Software Eng. 31(6): 429-445 (2005)

TeamTracks

- Shows source code navigation patterns of team
  - Related Items – most frequently visited either just before or after the selected item
  - Favorite Classes – hide less frequently used
- Deployed for real use – 5 developers for 3 weeks
- Successful, but usability issues, seemed most useful for newcomers

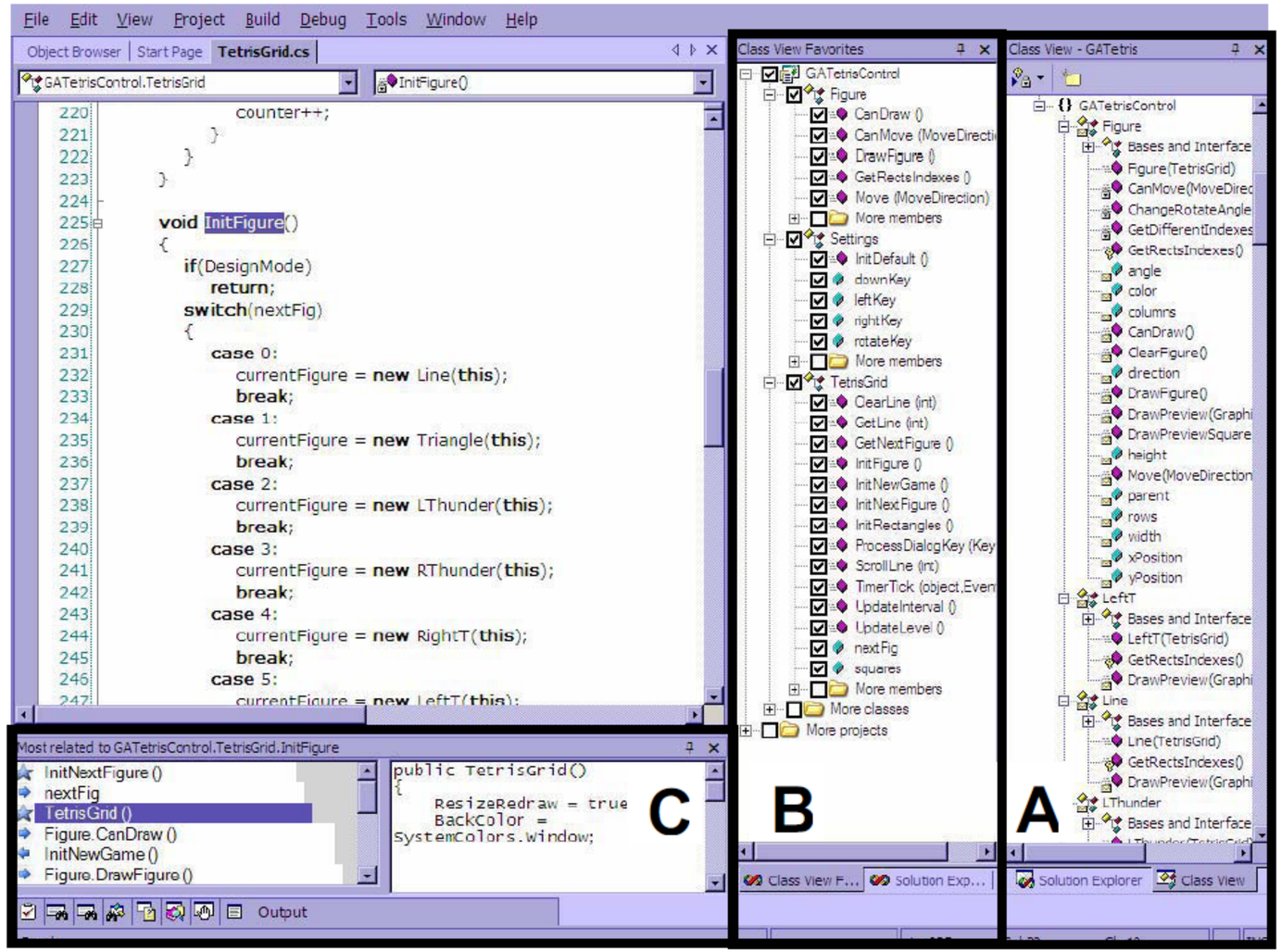


Figure 1. Screen shot with (A) the typical class view, (B) our Class View Favorites, and (C) our Related Items.

R. DeLine, M. Czerwinski and G. Robertson, "Easing program comprehension by sharing navigation data," Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05), 2005, pp. 241-248.

Task context navigation

- Make it easier to navigate back and forth between task context elements
- Make it easier to resume tasks by redisplaying task context

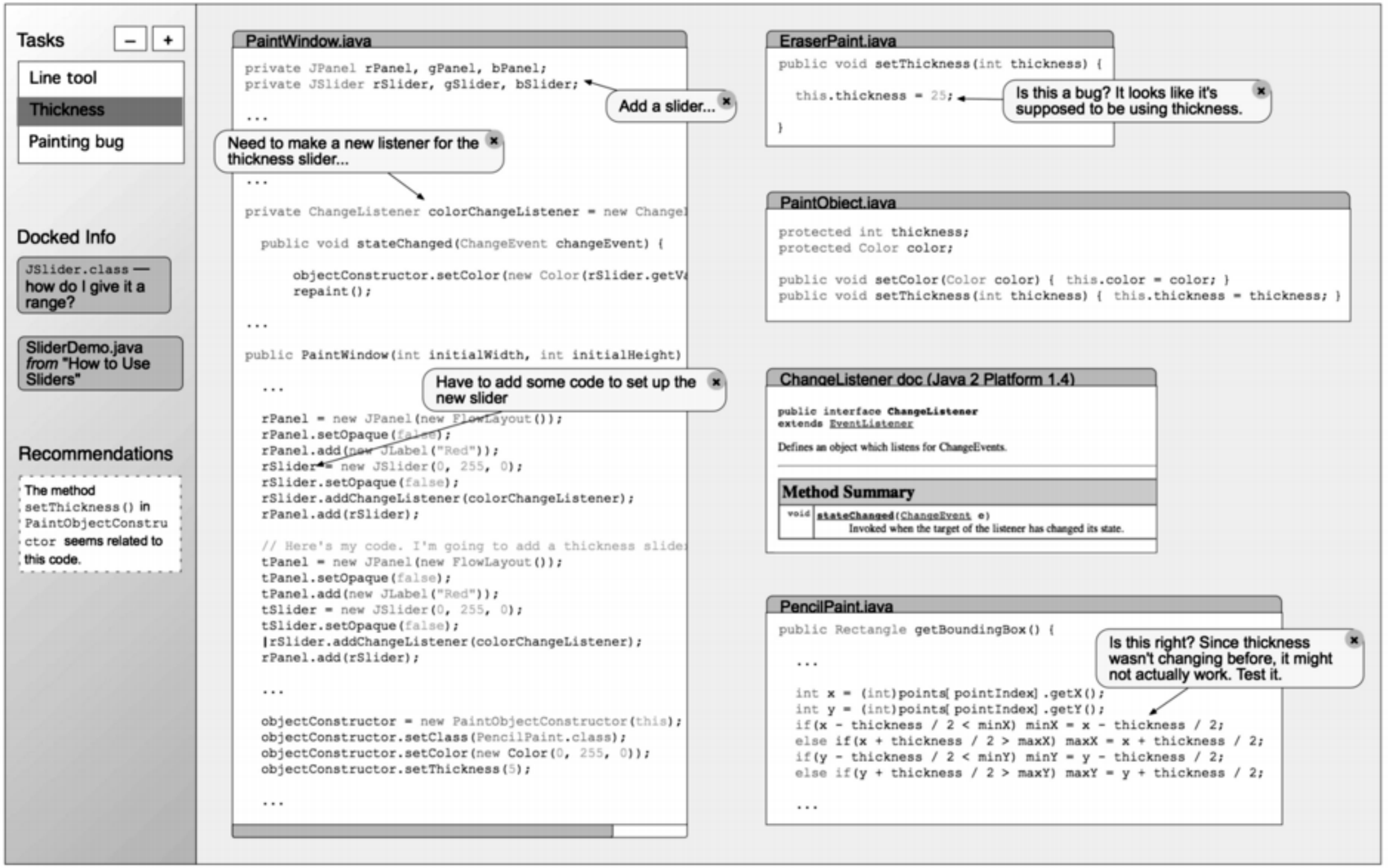
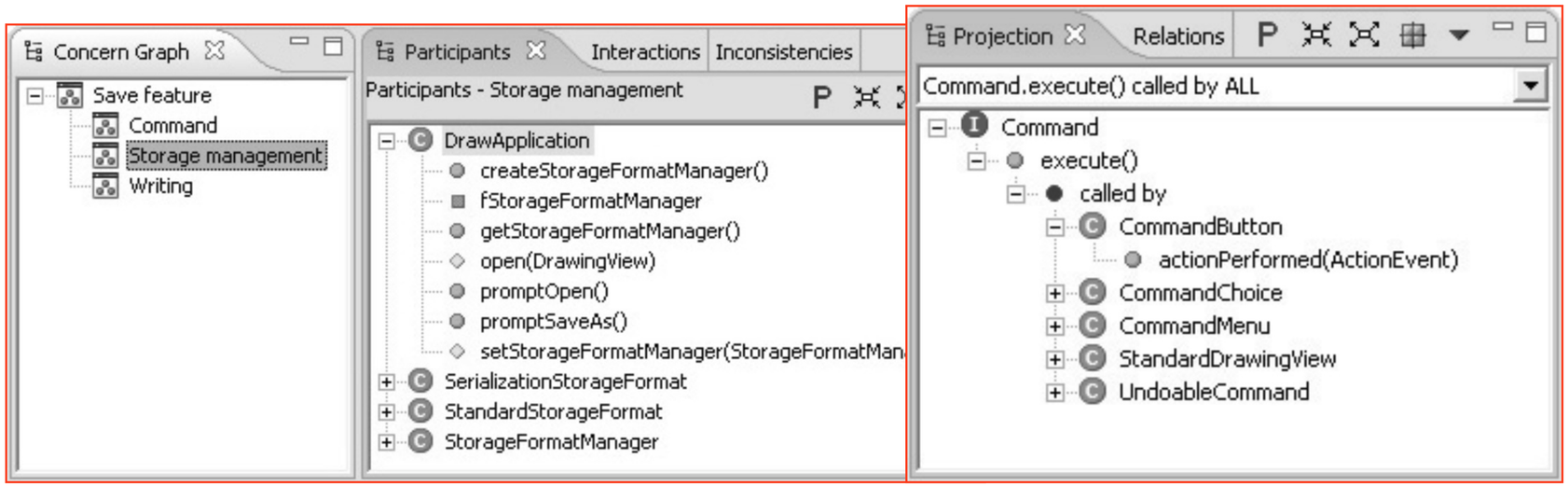


Fig. 7. The 50 lines of code and other information that developer B indicated as relevant, portrayed in a mockup of a workspace that help developers collect relevant information for a task in one place, independent of the structure of a program.

Andrew J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. 2006. An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks. IEEE Trans. Softw. Eng. 32, 12 (December 2006), 971-987.

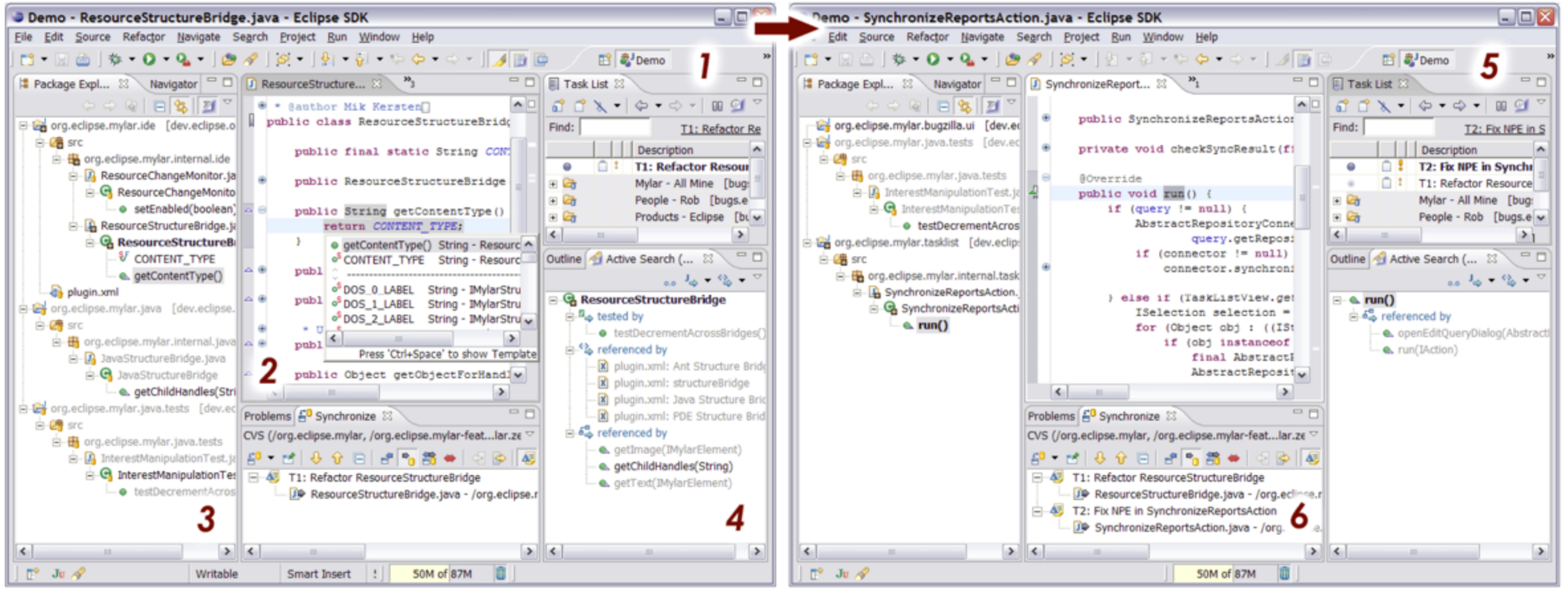
Concern Graphs

- Abstract (formal) model that describe which parts of the source code are relevant to different concerns
- FEAT tool builds concern graphs “semi-automatically”
- Shows only code relevant to the selected concern
- User-specified or detected using intra-concern analysis
- User can make queries



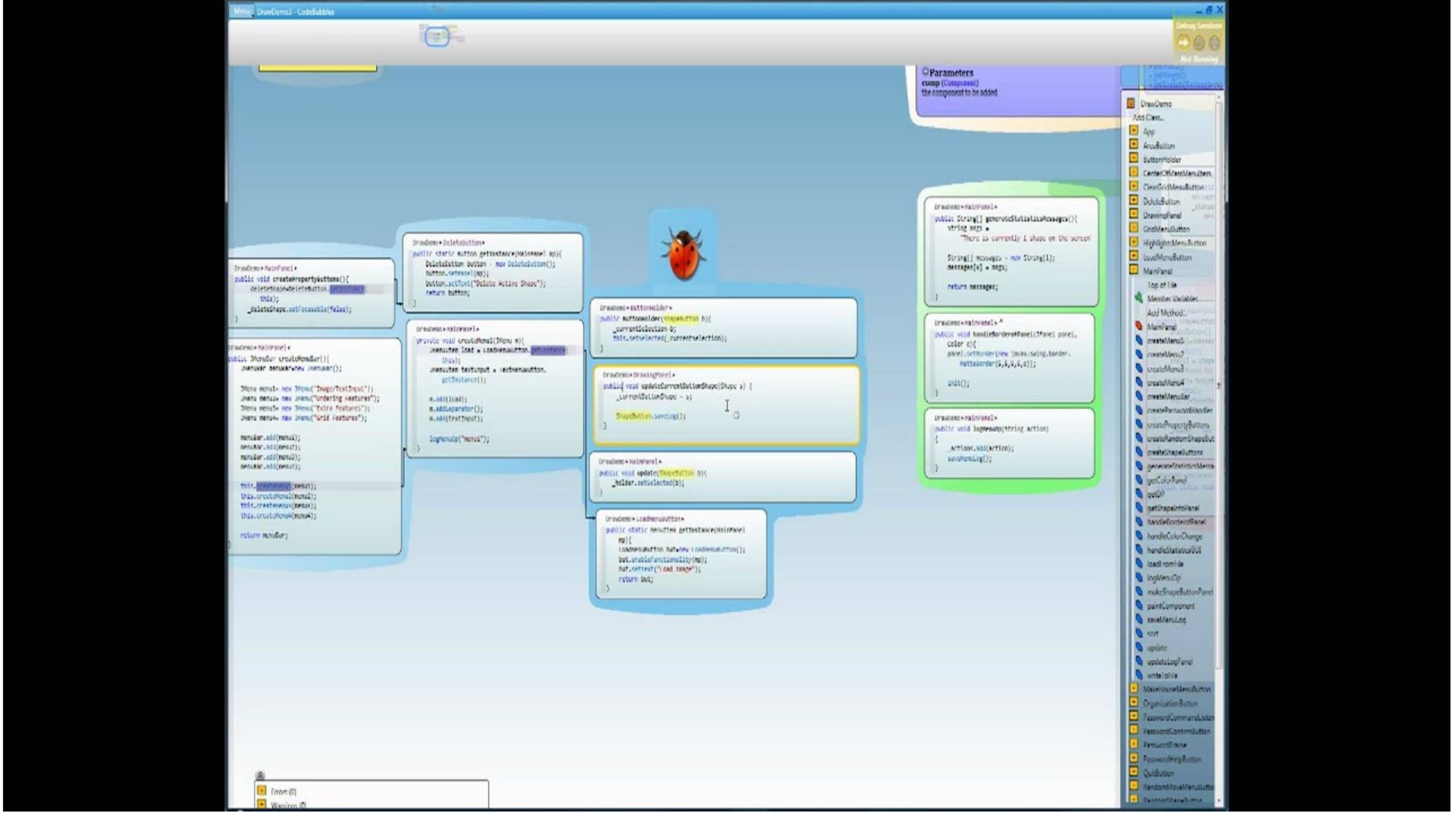
Martin P. Robillard and Gail C. Murphy. 2007. Representing concerns in source code. ACM Trans. Softw. Eng. Methodol. 16, 1, Article 3 (February 2007).

Mylar

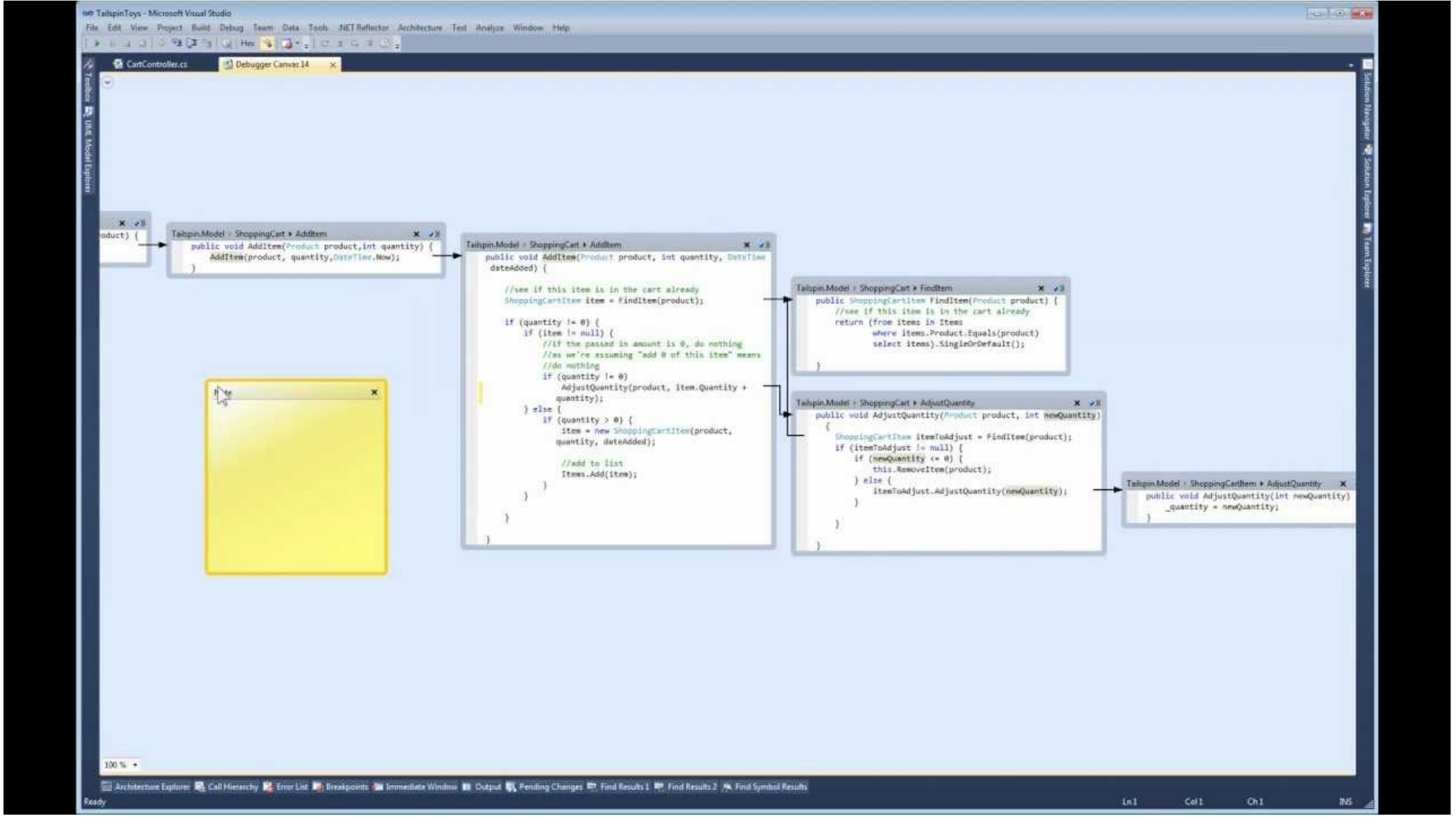


1 – task list
3 – package explorer filters to show what relevant to this task
Most relevant are bold
4 – active search shows what might be relevant
5 – switch to different task

Mik Kersten and Gail C. Murphy. 2006. Using task context to improve programmer productivity. International symposium on Foundations of software engineering, 1-11.



Andrew Bragdon, Robert Zeleznik, Steven P. Reiss, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola, Jr.. 2010. Code bubbles: a working set-based interface for code understanding and maintenance. Conference on Human Factors in Computing Systems (CHI '10), 2503-2512.



R. DeLine, A. Bragdon, K. Rowan, J. Jacobsen and S. P. Reiss, 2012. "Debugger Canvas: Industrial experience with the code bubbles paradigm," International Conference on Software Engineering, 1064-1073.

Use in practice

- Debugger Canvas offered as extension to Visual Studio https://marketplace.visualstudio.com/items?itemName=DebuggerCanvasTeam.DebuggerCanvas
- Mylar —> Mylyn, part of default Eclipse
- Mylyn —> commercial

Instant access to documents, web pages and code

With Tasktop, you can indicate when you start working on a task with a single-click. Tasktop then automatically tracks the web pages and desktop documents you work with and builds a model of how relevant each is for that task. Tasktop uses the model to show you just those pages and documents that are needed for a task. Switching to another task is as simple as clicking on that task. Tasktop Dev will show you *just* the information needed for that task. Think of the time you will save only working with the information you need! If you are using Tasktop Dev within your IDE, Tasktop Dev will focus the views in your IDE to show you just the code, web pages and documents that are needed for a task.

Keep on top of your bugs without leaving your IDE

Tasktop Dev inherits Mylyn's capabilities for rich bug editing in Eclipse, making the tasks from your issue tracker available in the IDE. With your task and its associated social comment thread in the IDE, you no longer have to spend time switching to a web browser, finding the appropriate task, switching back to the IDE, and so on. All of those clicks and applications switches disappear, saving time and keeping you close to the code. When changes occur to tasks on which you work, notifications appear right in your task list in Eclipse, saving you the need to check other applications and allowing you to respond seamlessly and as a part of your workflow. This capability is only available in Tasktop Dev for Eclipse and Visual Studio.

https://www.tasktop.com/tasktop-dev

Results from Debugger Canvas deployment

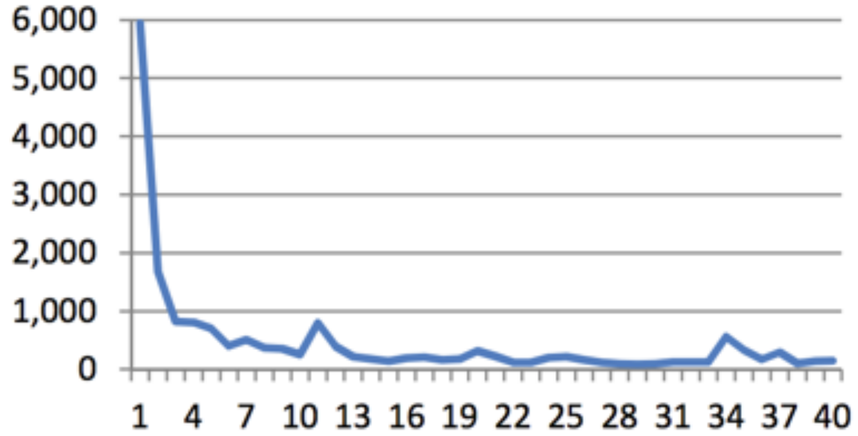


Figure 5: Number of unique downloads per week, after the initial release on 13 June 2011.

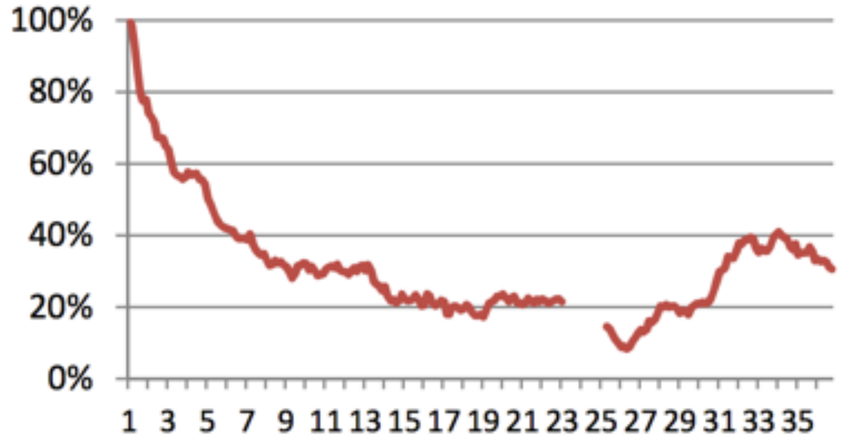


Figure 6. Users per day who step into a code bubble at least once, as a percentage of usage on the first day. (The gap is due to missing data.)

R. DeLine, A. Bragdon, K. Rowan, J. Jacobsen and S. P. Reiss, 2012. ”Debugger Canvas: Industrial experience with the code bubbles paradigm," International Conference on Software Engineering, 1064-1073.

Perceptions of debugger canvas

Table 2: Reasons why 19 surveyed users stopped using Debugger Canvas. Each respondent may report multiple reasons.

Reason to abandon	#Respondents	Type
Editing not discovered	4	Usability
Bugs	4	Bugs
Performance	4	Bugs
Doesn't support my platform	3	Other
Screen too small	2	Useful
Data tips bug	2	Bugs
Wants features	2	Utility
Concept didn't work for me	2	Utility
Want to resize bubbles	1	Utility
Instruction pointer update bugs	1	Bugs
Navigation not discovered	1	Usability
On demand not discovered	1	Usability
Sum	27	

R. DeLine, A. Bragdon, K. Rowan, J. Jacobsen and S. P. Reiss, 2012. "Debugger Canvas: Industrial experience with the code bubbles paradigm," International Conference on Software Engineering, 1064-1073.

Useful when

“I often have to debug several layers on our side from the UI, via middle tier to the data layer. It often gets confusing to go into the deeper layer. This is where the canvas helps, you hit a breakpoint here and can see the stack trace as you step through the layers. This helps us debug things much faster.”

“I was working on a large project for only a week. There was a huge ramp up, of course, and Debugger Canvas was invaluable for stepping into the code to see what was going on.”

“With a really large code base that you are not familiar with it is really handy. It helps wrap your head around other people's code. That kind of visualization really helps to follow code as it crosses different classes and projects. Go-to-definition and using Reflector is just too cumbersome to navigate through all that code.”

R. DeLine, A. Bragdon, K. Rowan, J. Jacobsen and S. P. Reiss, 2012. ”Debugger Canvas: Industrial experience with the code bubbles paradigm," International Conference on Software Engineering, 1064-1073.

Not useful when

For a "normal" project it isn't worth the hassle with performance.

I don't always want to get into the canvas. When I'm debugging something small: for example - Did the parameter get here? Then it doesn't warrant opening up the canvas.

Sometimes the fix that I need to do involves code that is not in the bubbles, but is in the same files, so I'd like to be able to get to the rest of the file easily.

I stop using it when I need to see definition of classes. I'm aware of the Go-to-definition feature, but I use ReSharper and lots of tools to navigate, so I find it easier to go back to the file in those cases.

I hit a breakpoint check the value of a private field.That's when seeing the rest of the file comes in handy.

R. DeLine, A. Bragdon, K. Rowan, J. Jacobsen and S. P. Reiss, 2012. "Debugger Canvas: Industrial experience with the code bubbles paradigm," International Conference on Software Engineering, 1064-1073.

10 min break

Tech Talks

In-Class Activity

- In groups of 2 or 3, write a reflecting on your experiences with code navigation tools.
  - Consider tools such as Go To Definition, Find in Files, Call Hierarchy, Debugger, Tabs
  - Write a strategy for each of the following common tasks reflecting how you would typically approach the problem
    - Find all of the methods involved in feature x.
    - Understanding what a method does and when it is called.
    - Manage a working set of files you are working on together for a commit
  - Based on your strategy for each task, reflect on challenges you experience. How might specific tools help address some of these challenges?
- Submission
  - Submit (1) pdf or doc with reflection and (2) source code through Blackboard. 1 submission per group. Due 7:00pm today.