A Field Study of How Developers Locate Features in Source Code

Kosta Damevski (Virginia Commonwealth University)
David Shepherd (ABB Inc)
Lori Pollock (University of Delaware)

*paper appears in the Journal of Empirical Software Engineering
volume 21, issue 2, pp 724-747

ICSE 2016
Austin, TX

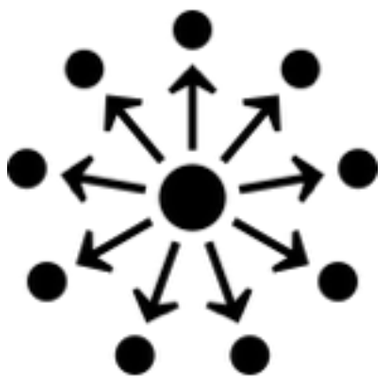
School of Engineering | Computer Science

Feature Location in the Field

- Feature Location = finding relevant code elements to perform a maintenance task
- Many lab studies, but relatively few field studies of feature location
  - Field studies' advantages:
    - ***realism***
    - ***scale***
    - ***no observational bias***

Our *Field* Study of F.L.

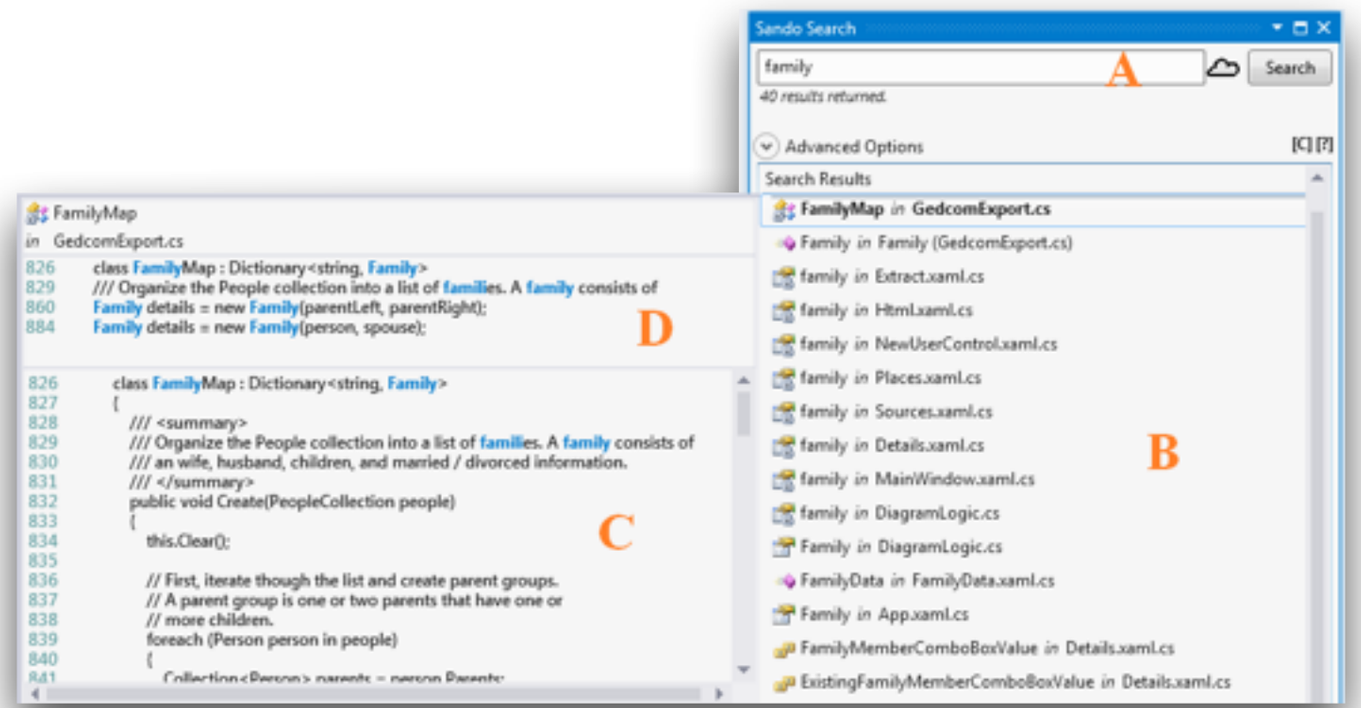
- Dual datasets:
  - ***Blaze*** dataset: 67 developers at ABB, Inc. for roughly two months
    - all clicks, key presses and IDE events in Visual Studio
  - ***Sando*** dataset*: 596 developers using a IR-based code search tool for roughly 1 year
    - statistics on query, corpus and result click



*after removing Sando users with only one query

Sando Code Search Tool

- *Sando* is an IR-based code search tool for Visual Studio
  - several year community open-source development project
  - free, open source, ~25K downloads



D. Shepherd, K. Damevski, B. Ropski, T. Fritz. " ***Sando: An Extensible Local Code Search Framework*** *". Proceedings of the 20th International Symposium on the Foundations of Software Engineering (FSE 2012), Raleigh, North Carolina, 2012*

Types of Findings

- From the two datasets, we report on:
  1. ***the use of code search tools (in Visual Studio)***
  2. ***multi-modal feature location***
     - code search *and* (structured navigation *or* debugging)
     - within a continuous section of time = a feature location session

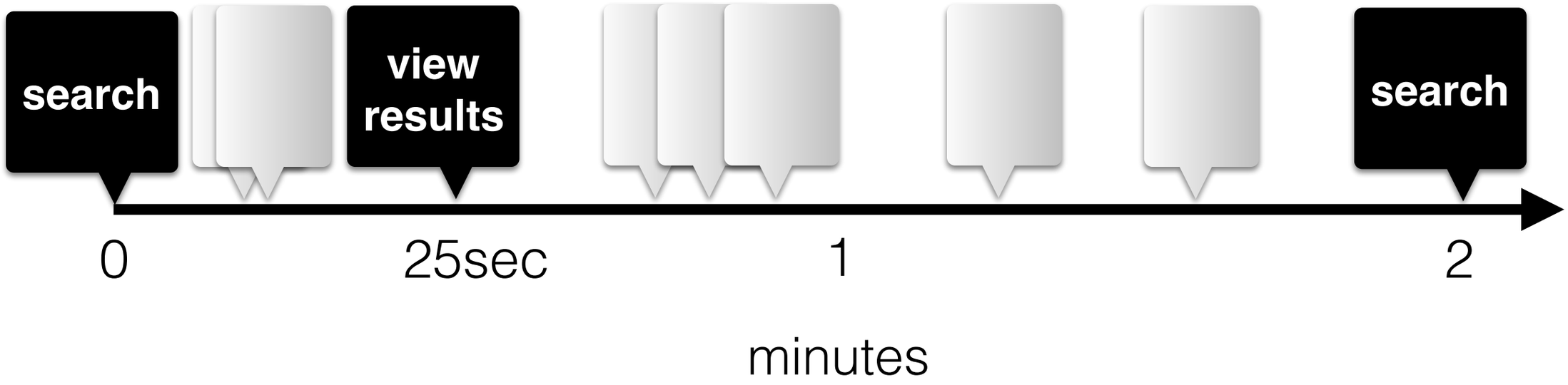
Interactions into Sessions

```
2013-11-18 14:50:03.000, dev_1, Debug.Start
2013-11-18 14:50:17.000, dev_1, View.File
2013-11-18 14:50:23.000, dev_1, View.OnChangeCaretLine
2013-11-18 14:50:24.000, dev_1, Debug.Debug Break Mode
2013-11-18 14:50:33.000, dev_1, View.Find in Files
2013-11-18 14:51:08.000, dev_1, View.Find Results 1
```

- Need to reason over developer tasks
  - BUT, our data is events
    - noisy
    - low-level
  - grouped data into feature location sessions, centered around behaviors of interest

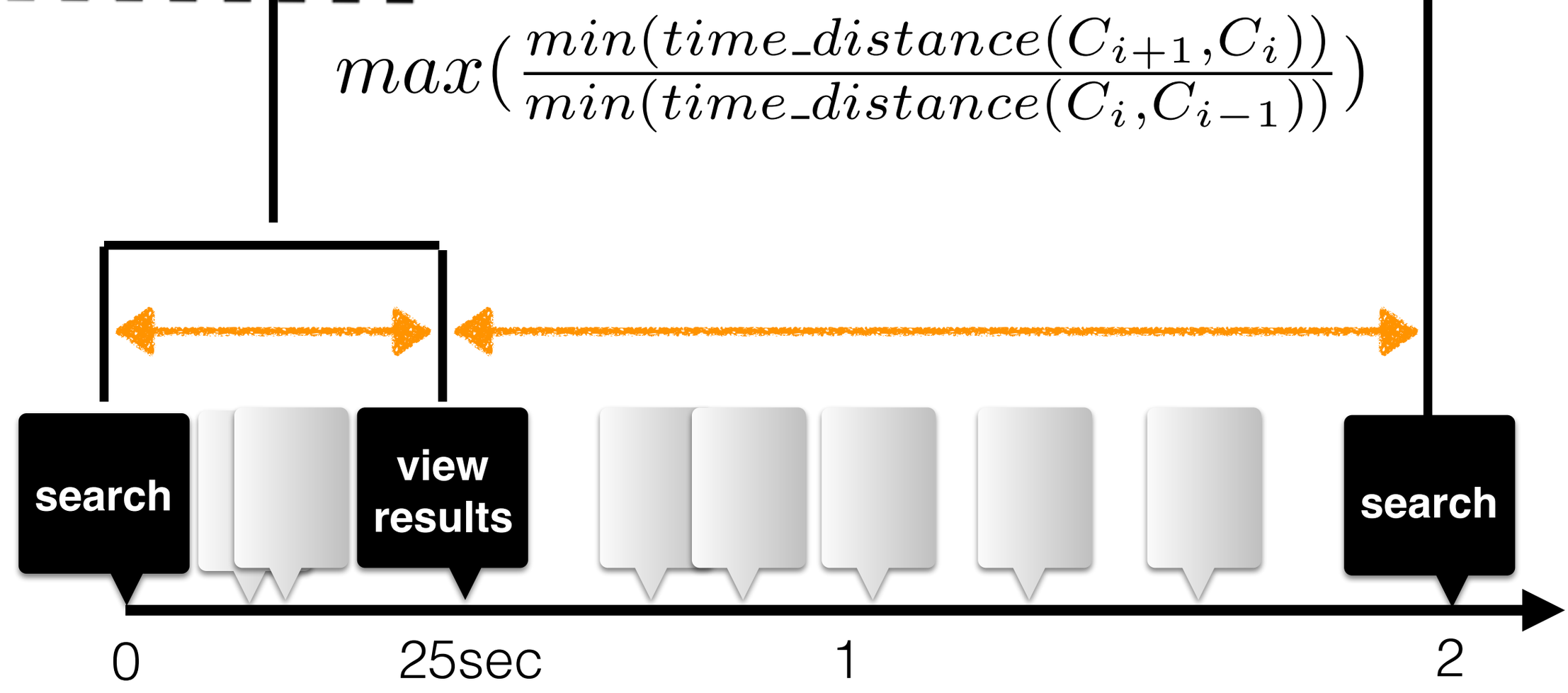
Sessionization

- Start with a set of ***key messages*** for a specific behavior
  - Use hierarchical agglomerative clustering with a natural cut
    - no need to choose an arbitrary session cut-off interval



Sessionization

$$max(\frac{min(time\_distance(C_{i+1}, C_i))}{min(time\_distance(C_i, C_{i-1}))})$$

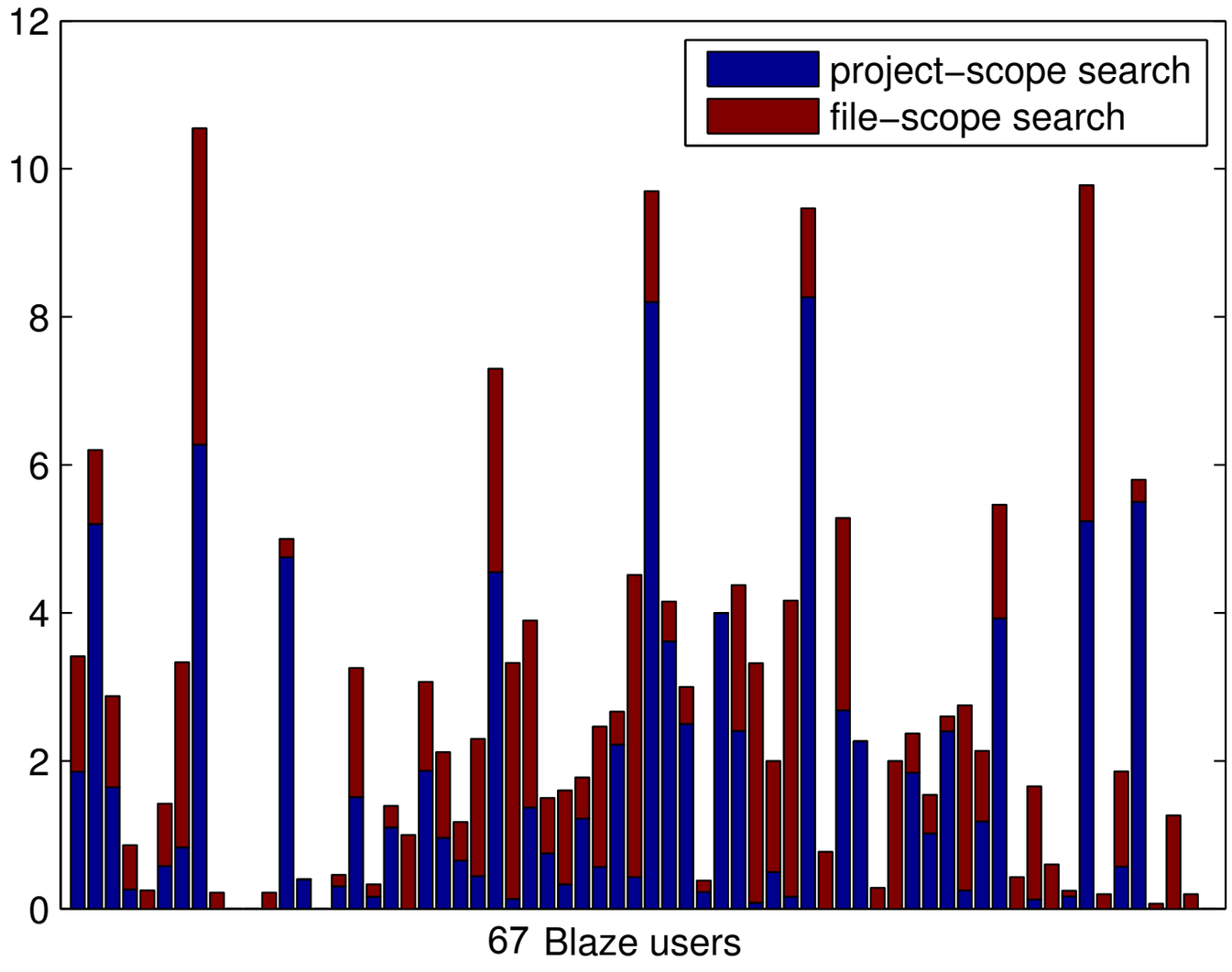


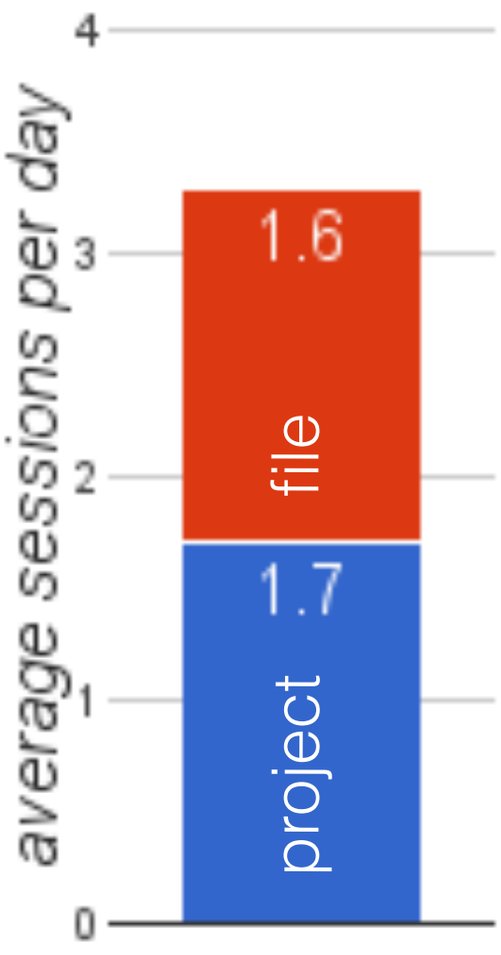
Code Search Tools in VS

- ***File Scope:***
  - *QuickFind*
    - ctrl+F

- ***Project Scope:***
  - *NavigateTo*
    - like *OpenType* for Eclipse
  - *Find in Files*
    - grep on entire project
  - *Sando*
    - IR-based tool

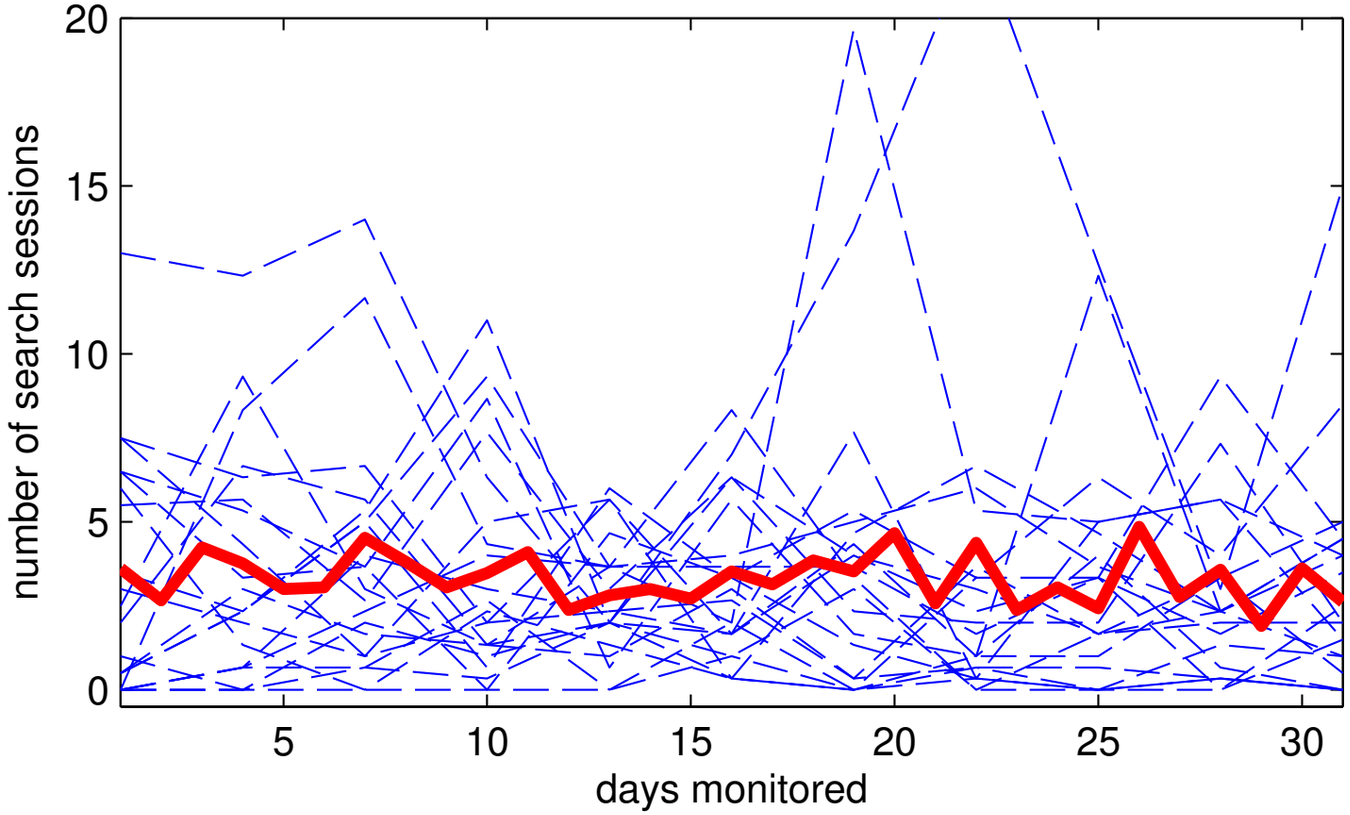
Code Search Sessions



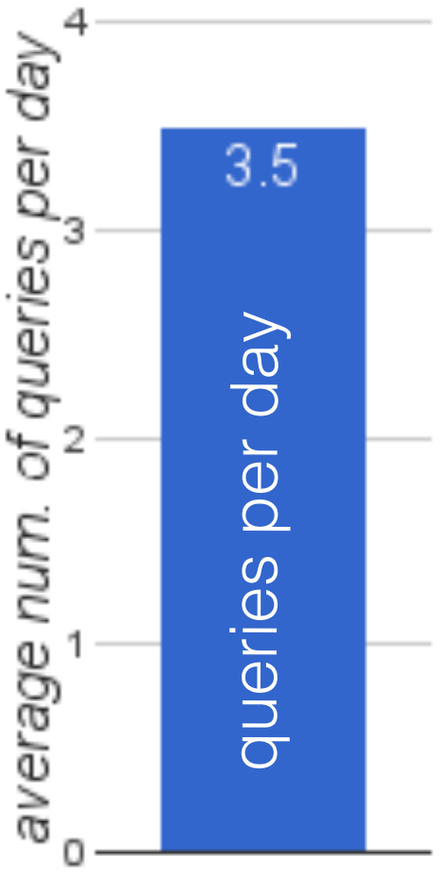


- Both project-scope and file-scope search are used frequently

Regularity of Code Search

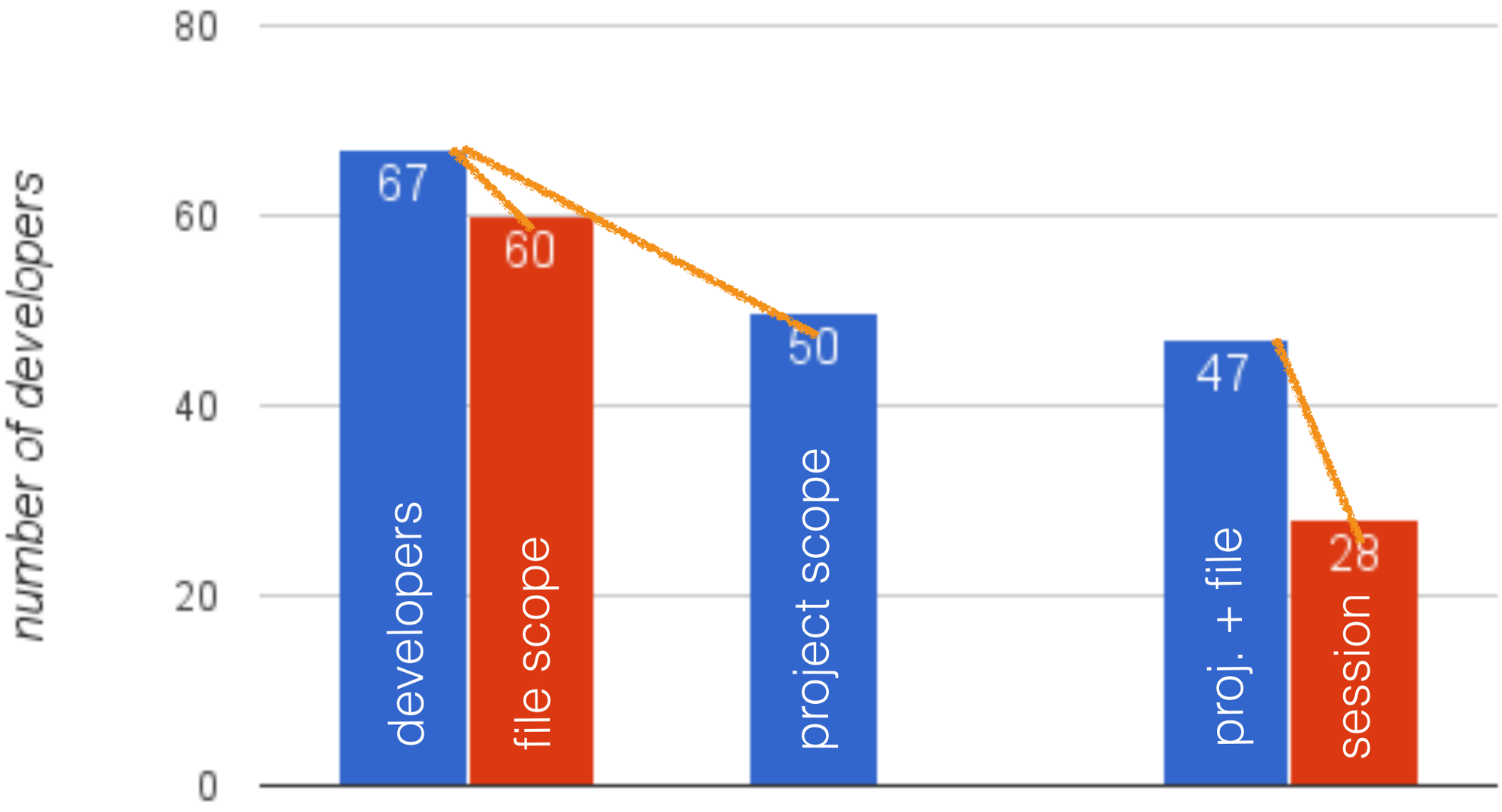


**graph represents average over a 3 day window*



- Code search tools are used fairly regularly (> 1 daily)

Code Search Use by Dev's



- Project scope + file scope sessions either followed a narrowing or an expanding pattern

Rarely Used *NavigateTo*

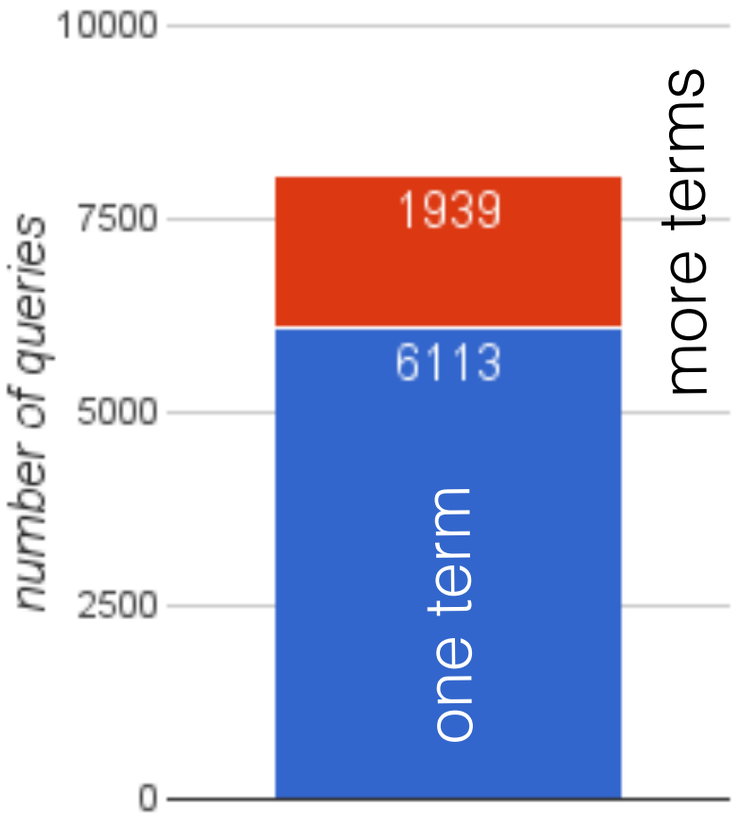
- Only 2/67 developers issued a query on NavigateTo
    - used other search tools before and after
- Studies based on Eclipse UDC dataset found that *OpenType* was one of the least used search tools [Murphy et al.,2006]

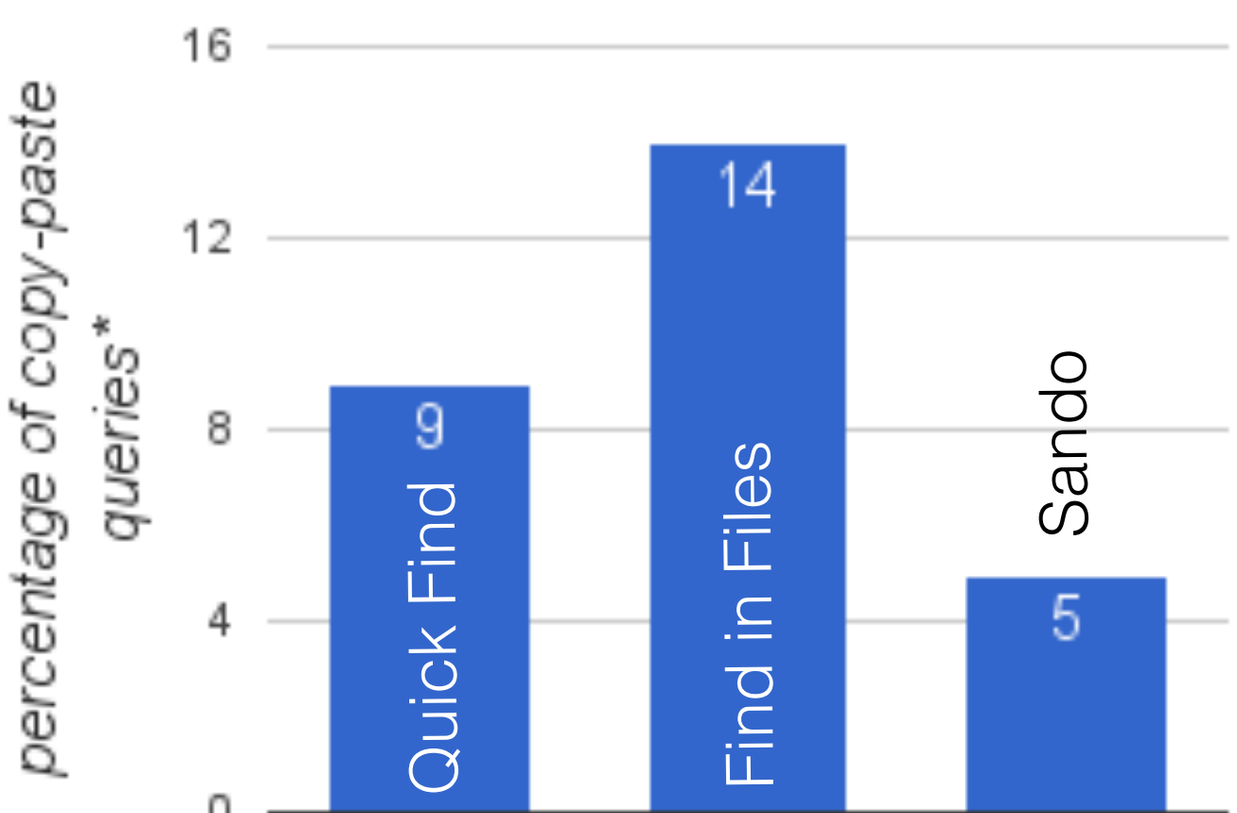
Find in Files vs. *Sando*

- IR-based code search tools (*Sando*) are aimed to be a replacement for string matching tools (*Find in Files*)
  - 1/3 developers never used *Find in Files* after their first *Sando* query
  - 2/3 developers using both tools used them interchangeably

**Sando didn't index JavaScript or VB*

Querying Behavior





- Developers issue one term queries and sometimes rely on copying from the code base to generate queries

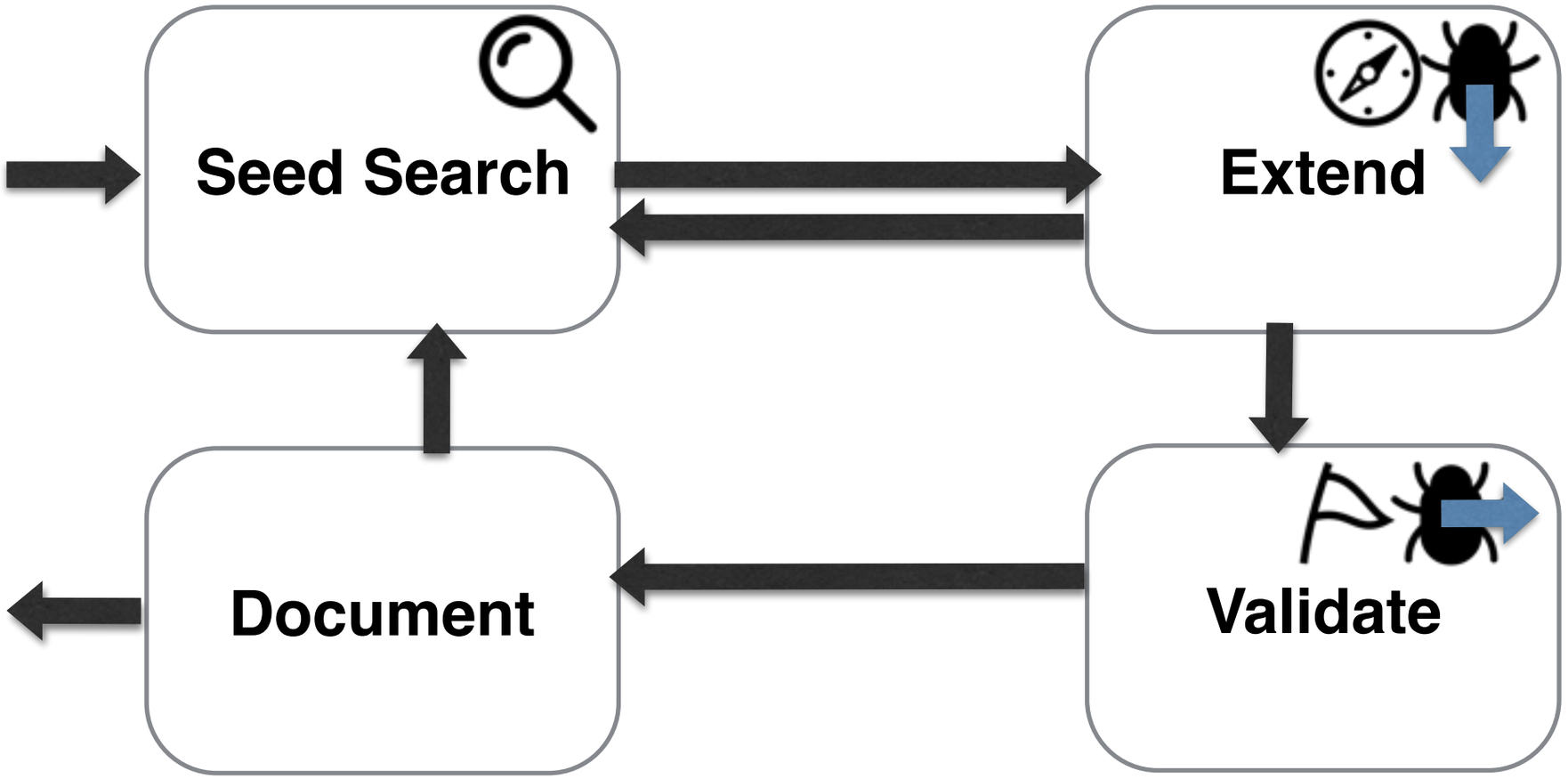
Query Reformulation

- 672 (from 8025) *Sando* queries (or 8.35%) were part of a reformulation sequence
  - predominantly by adding one term (~ 25% of reformulated queries)
  - some by removing one (or more) terms (~ 5% of reformulated queries)

Implications of Code Search Tool Study

- People use code search regularly; if we improve it we can effect their professional lives
- Flexible code search tools
    - many lookup (not exploratory) queries
    - lack of flexibility could explain why *NavigateTo* was used infrequently
- Information foraging (berry picking) model of code search seems to occur often

Multi-Modal Feature Location

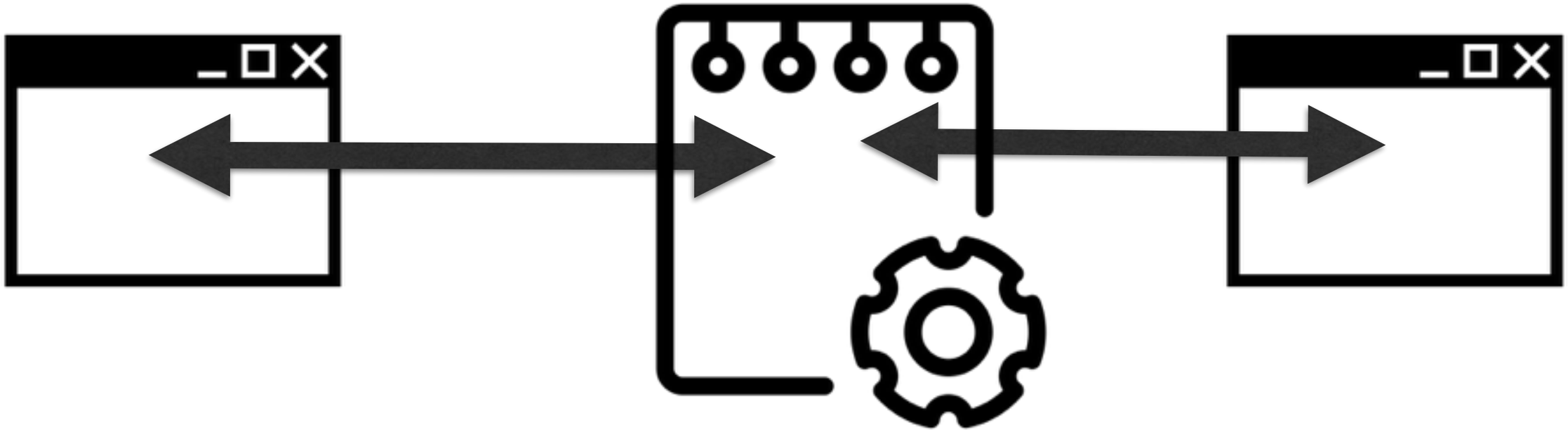


- Subset of F.L. model based on lab study by Wang et al.,“An Exploratory Study of Feature Location Process” ICSM 2011.

Repetitive Tool Use in F.L.

- Found 206 multi-modal sessions in dataset
- Alternating modalities in 41/206 or 20% of multimodal feature location sessions
  - e.g. search -> debug -> search again
  - majority of sessions alternated between search and structured navigation

Implications of Multi-Modal Feature Location Study



- Task context can help with using multiple modalities in the IDE

Summary

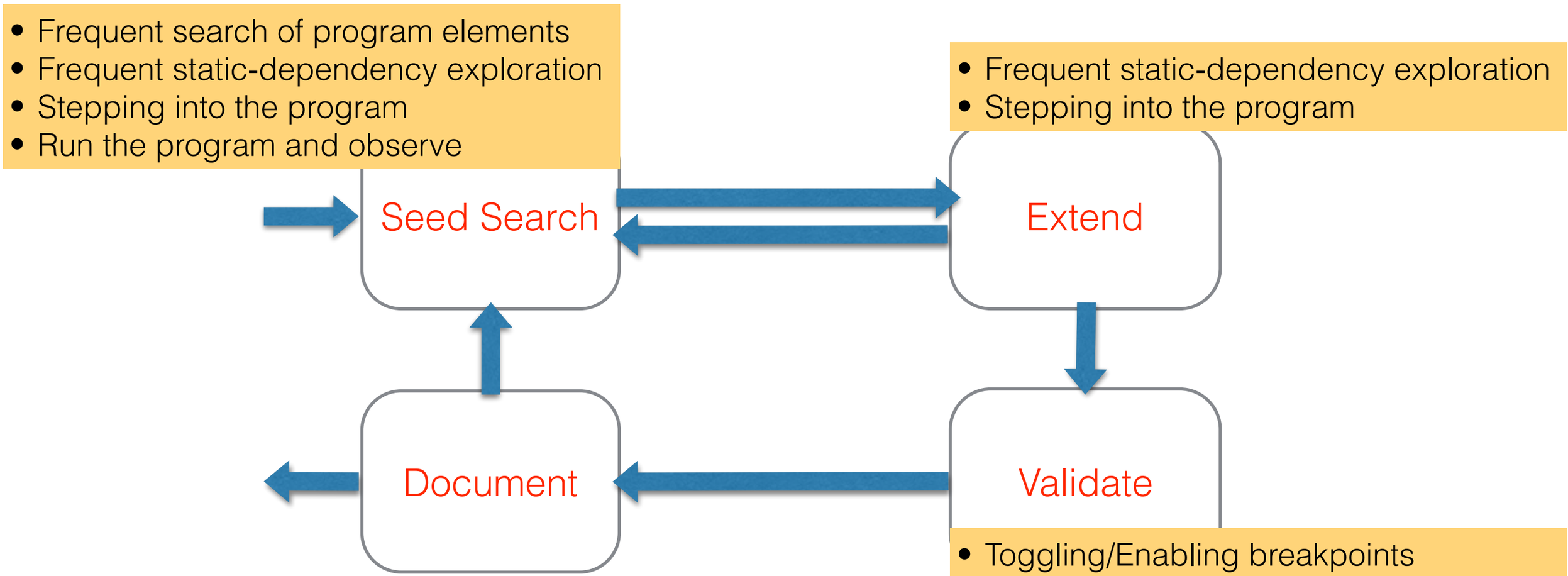
- ***More evidence*** that:
    - Developers use code search tools often
    - Queries are often short and commonly reformulated
    - Navigating program structure commonly follows code search
- (Relatively) ***new evidence*** that:
    - Some queries are created via copy and paste from code
    - Developers tend to repeatedly switch between different feature location modalities

Thanks!

QUESTIONS?

damevski@acm.org

Feature Location Process



- figure reproduced from Wang et al.,“An Explora
  Location Process” ICSM 2011.
    - analysis was based on 76 hours of full-screen videos of 38 developers' work on 12 feature-location tasks on four subject systems”