information needs in collocated software development teams
software developers...

understand dependencies
[de Souza 03][Ko 2006]

have a lot of questions
[Sillito 2006]

maintain awareness [Gutwin 2004]

interact with coworkers [Perry 94]

talk to experts [McDonald 1998][Hertzum 2002]

get interrupted a lot
[Perlow 99][Gonzalez 05]
what is all of this information?
research questions

what information is important to developers’ work?

where do they find it?

what prevents them from acquiring it?

we answered these by watching developers work.
findings

developers looked for info about design and program behavior, but often couldn’t find it.

this was because such info, if stored anywhere, was often stored only in their coworkers’ minds or difficult to search repositories of documents.

because such info was inaccessible, developers become blocked and interrupted coworkers, leading to work fragmentation.
outline

introduction

study method

activities and task switching

information needs

limitations and discussion
250 devs selected from the corporate address book.
55 responded. We asked, one at a time, if we could visit.
After visiting 17 developers (over 25 hours), we had seen enough redundancy to stop.
observations

25 hours of coding and bug fixing, in the role of “new hires”

357 pages of handwritten notes (other recording was too invasive)

4,231 events in an spreadsheet
outline

introduction

study method

activities and task switching

information needs

limitations and discussion
8 distinct activities

writing code
submitting code
triaging bugs
reproducing a failure
understanding behavior
reasoning about design
maintaining awareness
non-work activity
9 reasons for switching tasks

- face to face conversation
- phone call
- instant message
- e-mail alerts
- bug report change alerts
- task avoidance
- getting blocked
- meetings
- task completion
the 17 developers’ work
median time between interruptions

interrupted about every ~5-10 minutes
median time between getting blocked

blocked about every ~10 minutes
outline

introduction

study method

activities and task switching

information needs

limitations and discussion
identifying information needs

what has my team been doing?

M: “Make any progress on that bug?”

E: “He’s asking me [in IM] if I’m working on the stress break.”

how do I coordinate these artifacts?

A: “Is it okay to call KillExtra before KillTempFiles()?”

N: “I’m trying to figure out how to get the right instance.”

source

from a coworker

from a coworker

from a coworker

from code
21 information needs

what have my coworkers been doing?
what code caused this program state?
have resources I depend on changed?
what code could have caused this behavior?
how do I use this data structure or function?
did I make any mistakes?
what is the program supposed to do?
in what situations does this failure occur?
is this problem worth fixing?
why was this code implemented this way?
what’s statically related to this code?
what are the implications of this change?
what does the failure look like?
how can I coordinate this with the other code?
did I follow my team’s conventions?

full details in the paper
<table>
<thead>
<tr>
<th>Need</th>
<th># people</th>
<th># of seeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>what have my coworkers been doing?</td>
<td>15</td>
<td>43</td>
</tr>
<tr>
<td>what code caused this program state?</td>
<td>11</td>
<td>31</td>
</tr>
<tr>
<td>have resources I depend on changed?</td>
<td>10</td>
<td>25</td>
</tr>
<tr>
<td>what code <em>could</em> have caused this behavior?</td>
<td>9</td>
<td>22</td>
</tr>
<tr>
<td>how do I use this data structure or function?</td>
<td>9</td>
<td>21</td>
</tr>
<tr>
<td>did I make any mistakes?</td>
<td>9</td>
<td>46</td>
</tr>
<tr>
<td>what is the program <em>supposed</em> to do?</td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>in what situations does this failure occur?</td>
<td>7</td>
<td>27</td>
</tr>
<tr>
<td>is this problem worth fixing?</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>why was this code implemented this way?</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>what’s statically related to this code?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>what are the implications of this change?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>what does the failure look like?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>how can I coordinate this with the other code?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>did I follow my team’s conventions?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6 most common, unsatisfied

<table>
<thead>
<tr>
<th>Question</th>
<th>% Unsatisfied</th>
<th>Max Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>what have my coworkers been doing?</td>
<td>61%</td>
<td>21 min</td>
</tr>
<tr>
<td>what code caused this program state?</td>
<td>24%</td>
<td>9 min</td>
</tr>
<tr>
<td>have resources I depend on changed?</td>
<td>36%</td>
<td>17 min</td>
</tr>
<tr>
<td>what code <em>could</em> have caused this behavior?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>how do I use this data structure of function?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>did I make any mistakes?</td>
<td>15%</td>
<td>21 min</td>
</tr>
<tr>
<td>what is the program <em>supposed</em> to do?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>in what situations does this failure occur?</td>
<td>41%</td>
<td>49 min</td>
</tr>
<tr>
<td>is this problem worth fixing?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>why was this code implemented this way?</td>
<td>44%</td>
<td>21 min</td>
</tr>
<tr>
<td>what’s statically related to this code?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>what are the implications of this change?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>what does the failure look like?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>how can I coordinate this with the other code?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>did I follow my team’s conventions?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
how have my resources changed?

10 seeks  ≤9 minutes  24% unsatisfied

how are my builds doing? is my test suite done running?
what code has changed? has that bug been fixed yet?

a developer is blocked on a task, and decides to figure out how code and other resources have been changing.

sources  use tools, check e-mail, ask a coworker...
in what situations does this occur?

7 seeks \( \leq 49 \) minutes 41\% unsatisfied

what step is missing? what’s different about my configuration and the report author’s?

a developer gets a bug report and needs to reproduce the problem, but the reproduction steps do not work.

sources read the reproduction steps, talk to the report author, start twiddling bits, give up...
what could have caused this behavior?

9 seeks \( \leq 17 \text{ minutes} \quad 36\% \text{ unsatisfied} \\

could it be widget X or perhaps sprocket Z? maybe its getting into bad state Y...

a developer has just started trying to understand a failure, but needs to guess what’s causing it in order to use a debugger.

sources: guess; look for hints in the report; ask a coworker; make it not fail and find the difference.
what *actually* caused this state?

11 seeks \( \leq 21 \) minutes 61\% unsatisfied

why did this variable get set to 5? how did we reach this method? why are we in this thread?

a developer is trying to fix a bug, but to know what to fix, has to find the parts of the program that were involved.

sources set up the debugger, set some breakpoints and try to trace it back manually...
why was this code written this way?

6 seeks \( \leq 21 \) minutes 44\% unsatisfied

what was the author’s intent? was this overlooked intentionally or by mistake?

a developer figures out what’s causing a failure, but to avoid breaking other dependencies, needs to know why the code was written the way it is:

sources it’s written down in a report or a notebook somewhere, or it’s in the author’s head.
what’s the program *supposed* to do?

7 seeks  \( \leq 21 \) minutes  15% unsatisfied

what did we decide at that last design meeting? did I write this down somewhere? maybe it’s in the design spec

a developer starts on a bug report, figures out what the program is doing, but doesn’t know the intended behavior.

sources is there a spec I can trust? I’ll ask the PM; we might have to have a meeting about this...
findings

developers looked for info about design and program behavior, but often couldn’t find it.

this was because such info, if stored anywhere, was often stored only in their coworkers’ minds or difficult to search repositories of documents.

because such info was inaccessible, developers become blocked and interrupted coworkers, leading to work fragmentation.
limitations

corporate culture, testing practices, developer expertise, process can vary (though they did vary in our sample)

many of our observations were during bug fixing

probably had biases in our observations and categorization

our presence may have influenced developers’ work

some info seeking may have been too subtle to observe
Implications

Tools ought to encourage developers to *create and share* design knowledge as part of their work.

Tools need to help developers search for design knowledge, *wherever it is stored*.

*but...*
future work

what design knowledge should developers write down?

how should developers document design rationale?

if developers wrote down “everything,” how much would be useful?

what kind of searches would help developers find design knowledge that is written down?
of the information that developers did find, how accurate was it?
info needs

what code caused this program state?
why was this code implemented this way?
in what situations does this failure occur?
what code could have caused this behavior?
how have resources I depend on changed?
what is the program supposed to do?
what have my coworkers been doing?
how do I use this data structure or function?
did I make any mistakes?
is this problem worth fixing?
what's statically related to this code?
what are the implications of this change?
what does the failure look like?
how can I coordinate this with the other code?
did I follow my team's conventions?
what is the purpose of this code?
is this a legitimate problem?
what changes are part of this submission?
how difficult will this problem be to fix?
what information was relevant to my task?

icse '07 papers addressing problem

✓
✓
✓
✓
✓
✓
✓
✓
✓
✓
✓
✓
✓
✓
✓
✓
✓

lots of open problems!

other problems identified in icse ‘07 papers

preventing errors with specifications
making testing more feasible
identifying code clones
analyzing programs to detect errors
cross cutting concerns and refactoring
simplifying software deployment
participation in OSS
thanks!

also to...

the Microsoft developers for their participation
the HIP and VIBE teams at Microsoft Research
and
all of the summer ’07 interns!

www.cs.cmu.edu/~ajko

research.microsoft.com/hip