Questions Programmers Ask During Software Evolution Tasks

PRESENTED BY – SIMRANDEEP SINGH

Outline

INTRODUCTION

MAIN FOCUS

CONTRIBUTIONS

RELATED WORK

STUDY APPROACH

COMPARISON

CATEGORIZATION OF QUESTIONS

LIMITATIONS & OPEN QUESTIONS

Introduction

- Many tools are available to help programmers working on change tasks.

- But very little is known about the kinds of questions programmers ask when evolving a codebase.

Research Questions (main focus)

1. What does a programmer need to know about a code base when performing a change task to a software system?

2. How does a programmer go about finding that information?

Contributions

1. A catalog and a categorization of 44 kinds of questions programmers ask.

2. An analysis of the process of answering questions which exposed important context for the questions.

Related Work - Question Analysis

Letovsky observes programmer activities by

He reports on five kinds of **Conjectures**

Related Work - kinds of conjectures

- **WHY ----**
  Questioning the role of a piece of code
- **HOW ----**
  About the method for accomplishing a goal
- **WHAT ----**
  What is a variable or function
- **Whether ----**
  Whether or not a routine serves a purpose
- **Discrepancy ----**
  Questioning perceived discrepancies

Related Work - 7 questions need to be answered

Based on the personal experience, Erdos and Sneed suggests that seven questions need to be answered for a programmer to maintain a program that is partially understood:

1. Where is a particular subroutine/procedure invoked?
2. What are the arguments and results of a given function?
3. How does control-flow reach a particular location?
4. Where is a particular variable set, used or queried?
5. Where is a particular variable declared?
6. Where is a particular data object accessed?
7. What are the inputs and outputs of a module?

Related Work - Tool based studies

Storey et al. carried out a user study focused on

- How program understanding tools enhance or change the way that programmers understand programs.
- In their study thirty participants used various research tools to solve program understanding tasks on a small system.
- Based on these results they suggest that tools should support multiple comprehension strategies (top-down and bottomup, for example) and should aim to reduce cognitive overhead during program exploration.
- In contrast to this work, the Storey et al. study does not attempt to analyze specifically what programmers need to understand.

Study Approach

- **Situation:** One or two programmers working on a change task
- **Gather data:** Conversations between pairs of programmers and individual programmers talking aloud
- **Analyze:** Identified questions asked, and worked to understand how they were answered

Comparison

	SETTING	PARTICIPANTS	SYSTEMS	TOOLS	TASKS
STUDY 1	LABORATORY	9 GRADUATE STUDENTS (N1-N9)	ArgoUML (newcomers to this system)	ECLIPSE	ASSIGNED..
STUDY 2	INDUSTRIAL	16 PROFESSIONAL PROGRAMMERS (E1-E16)	VARIOUS SYSTEMS	VARIOUS..	SELECTED BY PARTICIPANTS

Study 1

- Pair Programming (information they looking for and particular actions)
- Total 12 sessions performed, with two participants working on assigned tasks using Eclipse (JDE) on single computer.
- Driver – who controls the mouse & keyboard (least experienced).
- Observer – Who is working with the driver (most experienced).

Session No.	Driver	Observer
1.1	N7	N3
1.2	N4	N1
1.3	N4	N7
1.4	N6	N4
1.5	N5	N3
1.6	N5	N2
1.7	N3	N1
1.8	N7	N5
1.9	N8	N6
1.10	N8	N2
1.11	N9	N2
1.12	N9	N6

Study 2

- Industrial Setup (16 professional programmers)
- Programmers working alone
- Participants were asked to select the tasks on which they worked earlier
- Participants describe their task selection and then about 30 minutes working on that task

SN	Part.	Primary Languages/Tools Used
2.1	E1	C++, TCL/Emacs, DDD, Virtual desktops
2.2	E2	C++, TCL/Emacs (split window)
2.3	E3	C#, XSLT/Visual Studio, BizTalk Orchestration
2.4	E4	C#/Visual Studio, BizTalk Orchestration
2.5	E5	C#, ASP/Visual Studio
2.6	E6,E7	C#, ASP/Visual Studio, NetMeeting
2.7	E8	Java/Netbeans
2.8	E9	SQL, MDX/Enterprise Manager, Query Analyzer,
2.9	E10	C++, Batch/Notepad, Visual Studio
2.10	E11	C/Proprietary loading and debugging tools
2.11	E12	C, C++/Visual Studio (two instances)
2.12	E13	HTML/UltraEdit, Proprietary Document Manager
2.13	E14	C/Emacs (split window)
2.14	E15	XML, Java/VIM (two instances)
2.15	E16	C/VI (two instances), GDB

Results

- 44 kinds of questions were asked by the participants.
- These questions are the generalized versions of the specific questions asked by the participants
- For example : N4 asked the question “How does [MAssociation] relate to [FigAssociation]?”
- This question was reported as “How are these types or objects related ”

44 kinds of questions asked

Which type represents this domain concept or this UI element or action?

Where in the code is the text in this error message or UI element?

Where is there any code involved in the implementation of this behavior?

Is there a precedent or exemplar for this?

Is there an entity named something like this in that unit (project, package or class, say)?

What are the parts of this type?

Which types is this type a part of?

Where does this type fit in the type hierarchy?

Does this type have any siblings in the type hierarchy?

Where is this field declared in the type hierarchy?

Who implements this interface or these abstract methods?

Where is this method called or type referenced?

When during the execution is this method called?

Where are instances of this class created?

Where is this variable or data structure being accessed?

What data can we access from this object?

What does the declaration or definition of this look like?

What are the arguments to this function?

What are the values of these arguments at runtime?

What data is being modified in this code?

How are instances of these types created and assembled?

How are these types or objects related? (whole-part)

How is this feature or concern (object ownership, UI control, etc) implemented?

What in this structure distinguishes these cases?

What is the behavior these types provide together and how is it distributed over the types?

What is the "correct" way to use or access this data structure?

How does this data structure look at runtime?

How can data be passed to (or accessed at) this point in the code?

How is control getting (from here to) here?

Why isn't control reaching this point in the code?

Which execution path is being taken in this case?

Under what circumstances is this method called or exception thrown?

What parts of this data structure are accessed in this code?

How does the system behavior vary over these types or cases?

What are the differences between these files or types?

What is the difference between these similar parts of the code (e.g., between sets of methods)?

What is the mapping between these UI types and these model types?

Where should this branch be inserted or how should this case be handled?

Where in the UI should this functionality be added?

To move this feature into this code what else needs to be moved?

How can we know this object has been created and initialized correctly?

What will be (or has been) the direct impact of this change?

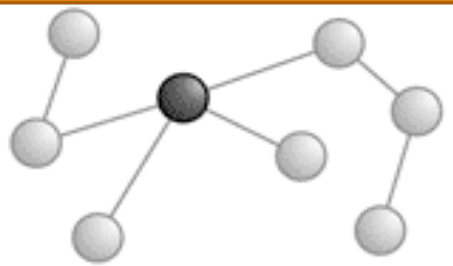
What will be the total impact of this change?

Will this completely solve the problem or provide the enhancement?

Categorization

Questions about : -

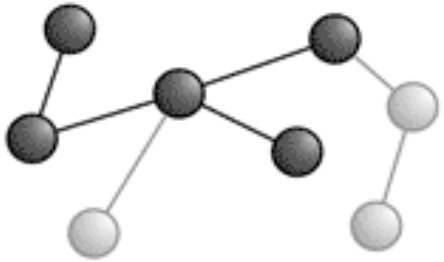
Discovering an initial entity in the graph	Given entity & other entities directly related to it	Understand No. of entities & relationships together	How Connected groups relate to each other and the rest of system
1	2	3	4



Finding focus points

5 kinds of questions.

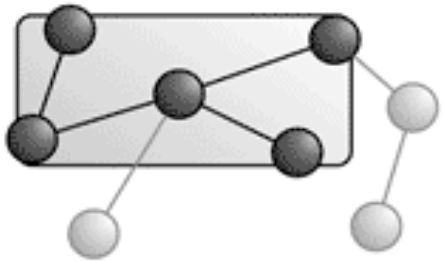
For example: Which type represents this domain concept?



Expanding focus points

15 kinds of questions.

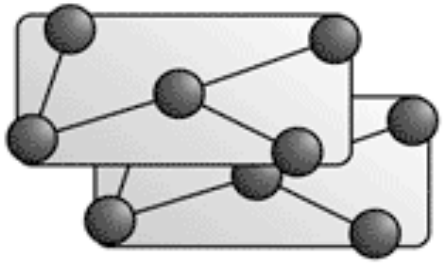
For example: Which types is this type a part of?



Understanding a subgraph

13 kinds of questions.

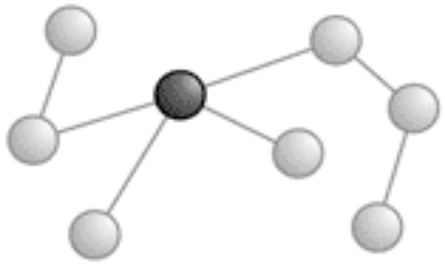
For example: What is the behavior these types provide together?



Questions over groups of subgraphs

11 kinds of questions.

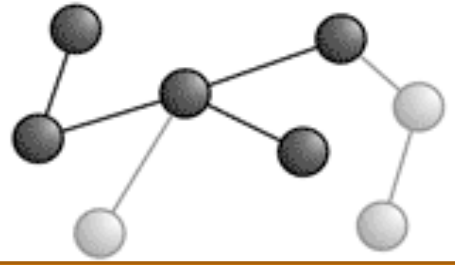
For example: What is the mapping between these UI types and these model types?



Finding focus points

5 kinds of questions.

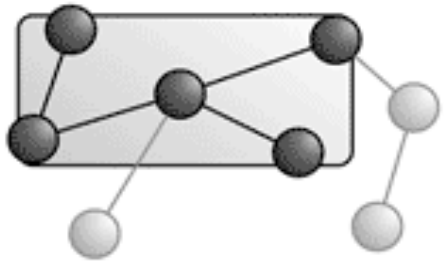
For example: Which type represents this domain concept?



Expanding focus points

15 kinds of questions.

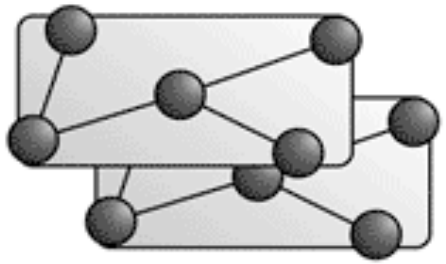
For example: Which types is this type a part of?



Understanding a subgraph

13 kinds of questions.

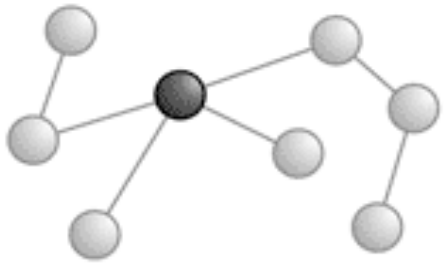
For example: What is the behavior these types provide together?



Questions over groups of subgraphs

11 kinds of questions.

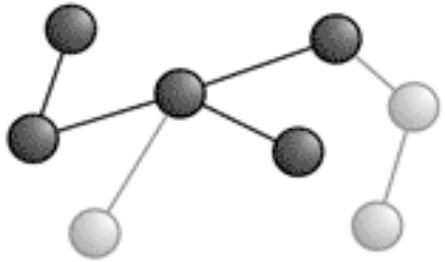
For example: What is the mapping between these UI types and these model types?



Finding focus points

5 kinds of questions.

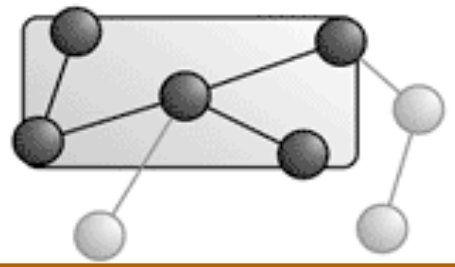
For example: Which type represents this domain concept?



Expanding focus points

15 kinds of questions.

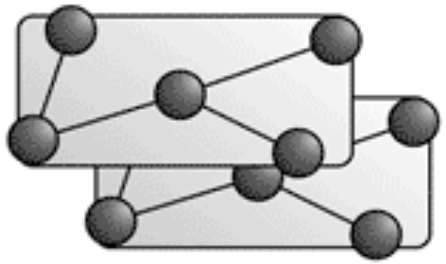
For example: Which types is this type a part of?



Understanding a subgraph

13 kinds of questions.

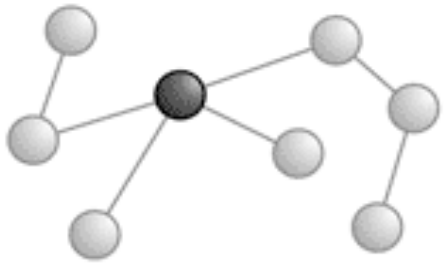
For example: What is the behavior these types provide together?



Questions over groups of subgraphs

11 kinds of questions.

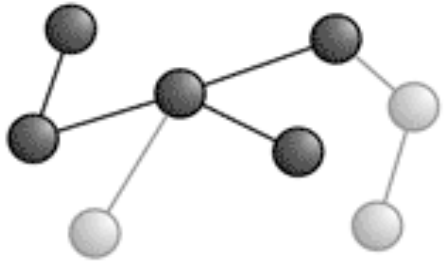
For example: What is the mapping between these UI types and these model types?



Finding focus points

5 kinds of questions.

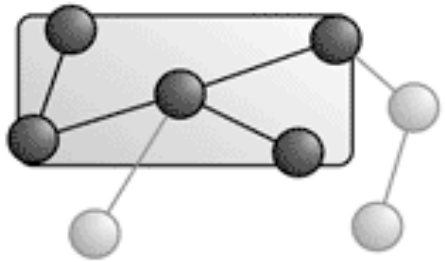
For example: Which type represents this domain concept?



Expanding focus points

15 kinds of questions.

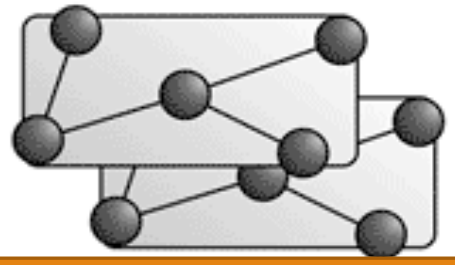
For example: Which types is this type a part of?



Understanding a subgraph

13 kinds of questions.

For example: What is the behavior these types provide together?

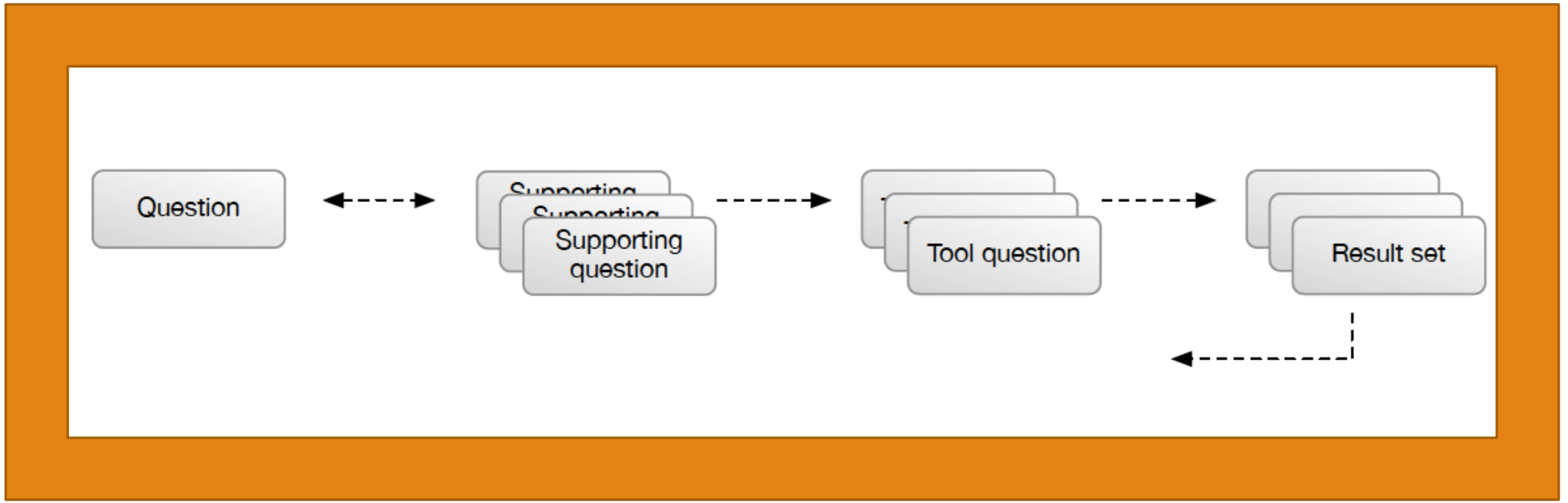


Questions over groups of subgraphs

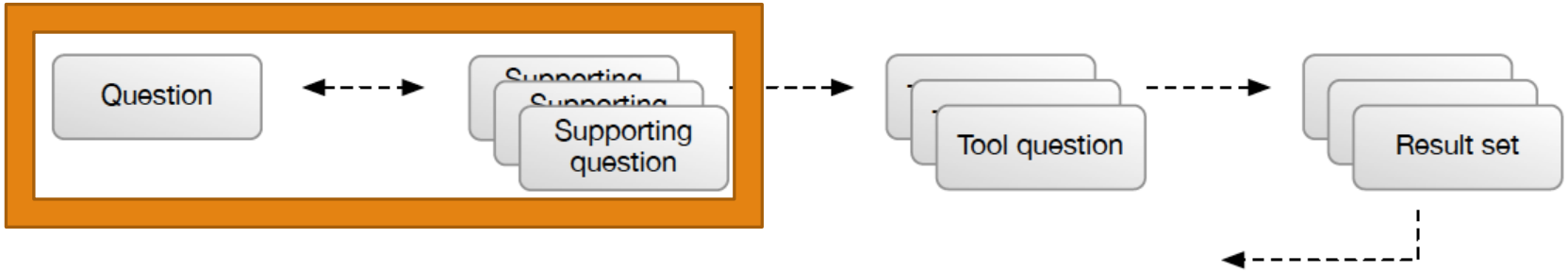
11 kinds of questions.

For example: What is the mapping between these UI types and these model types?

Answering Questions



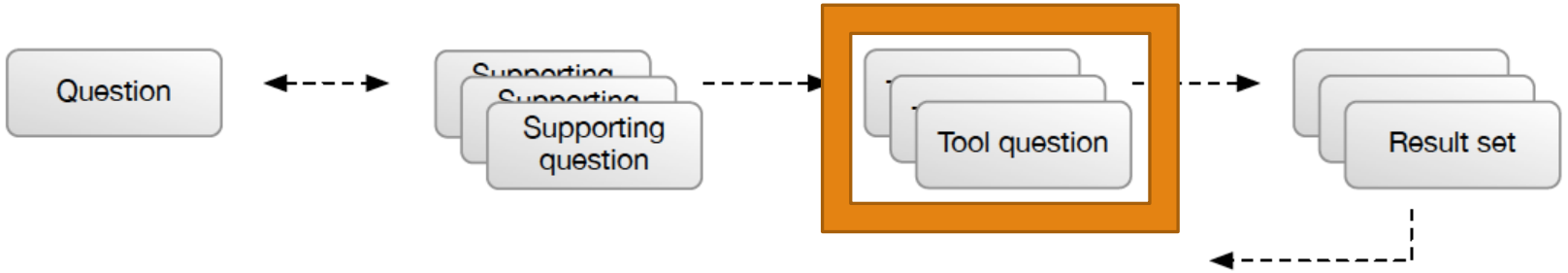
Relationship between questions



For example, questions and sub-questions.

“Trying to take my questions and filter those down to something meaningful where I could take a next step.” [N4].

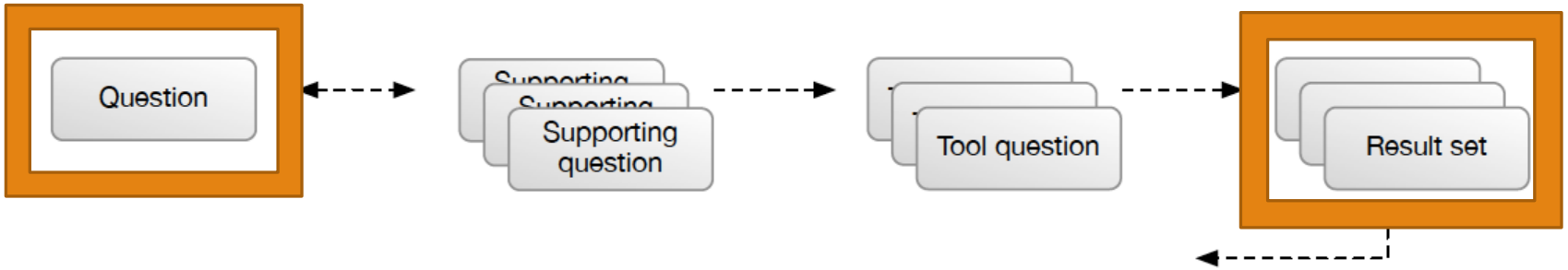
Questions and Tools



Mismatch between tools and questions and non-optimal choice of tools.

“Which classes have MEvents as fields?” [N3]

From results to answers



Multiple result sets, often noisy relative to the intended question. Often these results are often presented separately (can't be seen together).

"I was starting to forget who was calling what, especially because there is only one search panel at a time that I can see" [N6]

"You go down a path to try to find out some information and it leads to a dead end and you got to start all over again." [E16]

Implications

- Most tools support only relatively low-level questions.

For example, tools generally only support questions about one entity and type of connection; while some of the questions articulated by our participants were at the level of groups of entities and connections of different types.

- The result sets from tools include many items irrelevant to the original questions

The use of multiple tools required participants to mentally piece together pieces of information from multiple (often noisy) result sets.

Limitations and open questions

Generalizable?

- Types of software change tasks
- Programming tools our participants used

Open questions

- Do questions change with experience?
- Which questions are most important for tools to support?
- Which strategies for answering questions are most effective in the long run?

Thank You