



# **JTourBus: Simplifying Program Understanding by Documentation that Provides Tours Through the Source Code**

Christopher Oezbek, Lutz Prechelt

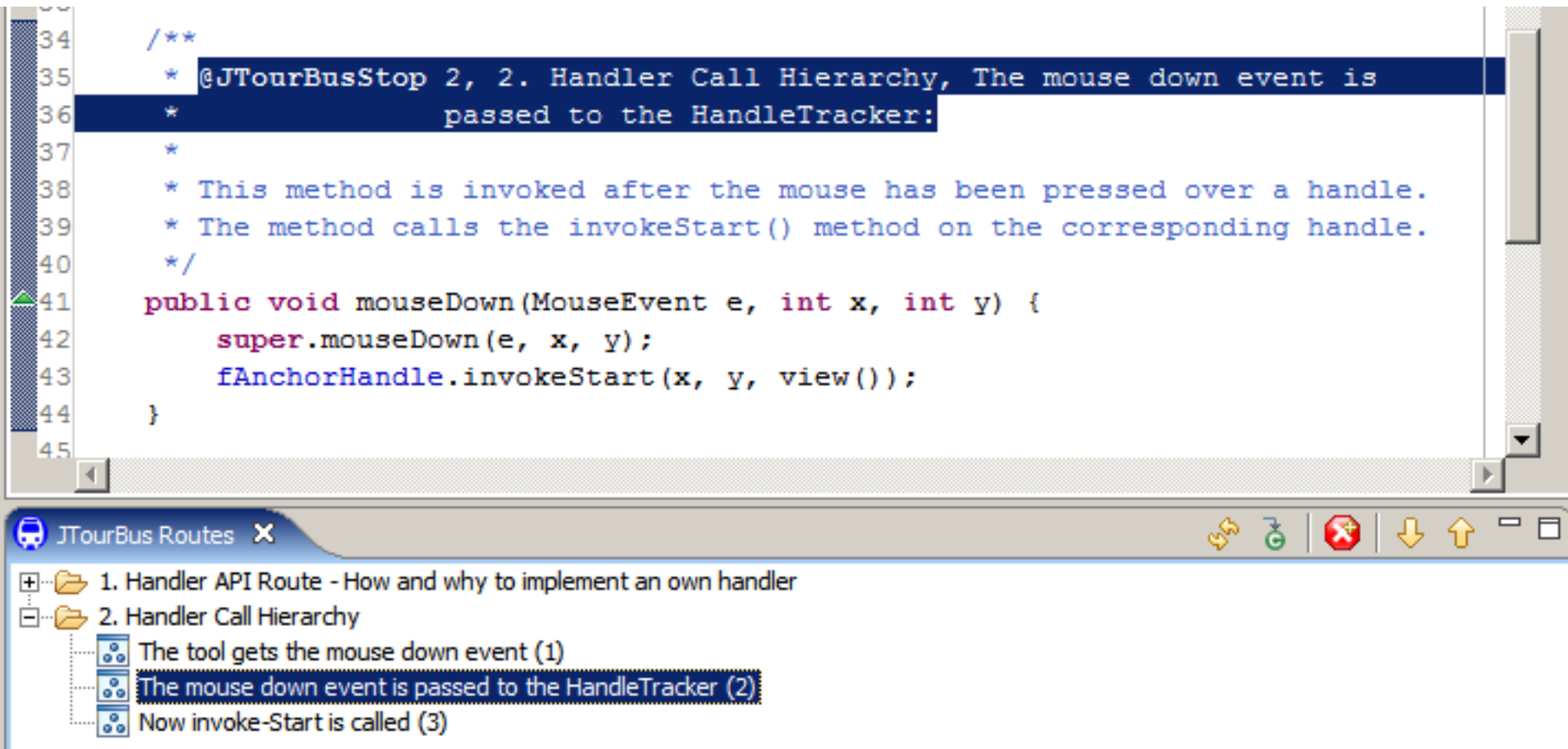
Institut für Informatik

Freie Universität Berlin

2007-10-02

- Given a medium-size system how do we best spend a limited amount of time to document the system for maintainability?
- Traditional answer:
  - Write a minimal design document
    - Present architecture and important concerns
- Our Answer:
  - Add high-level documentation as comments
  - Provide a tour through the code

- Mark interesting code locations with **@JTourBusStop**
- Use tool support to find these tags conveniently



The screenshot displays a code editor window with the following Java code:

```
34  /**
35   * @JTourBusStop 2, 2. Handler Call Hierarchy, The mouse down event is
36   *   passed to the HandleTracker:
37   *
38   * This method is invoked after the mouse has been pressed over a handle.
39   * The method calls the invokeStart() method on the corresponding handle.
40   */
41  public void mouseDown(MouseEvent e, int x, int y) {
42      super.mouseDown(e, x, y);
43      fAnchorHandle.invokeStart(x, y, view());
44  }
45
```

Below the code editor, a tool window titled "JTourBus Routes" is open. It shows a call hierarchy for the method `mouseDown`. The hierarchy is as follows:

- 1. Handler API Route - How and why to implement an own handler
- 2. Handler Call Hierarchy
  - The tool gets the mouse down event (1)
  - The mouse down event is passed to the HandleTracker (2)**
  - Now invoke-Start is called (3)



- Advantages

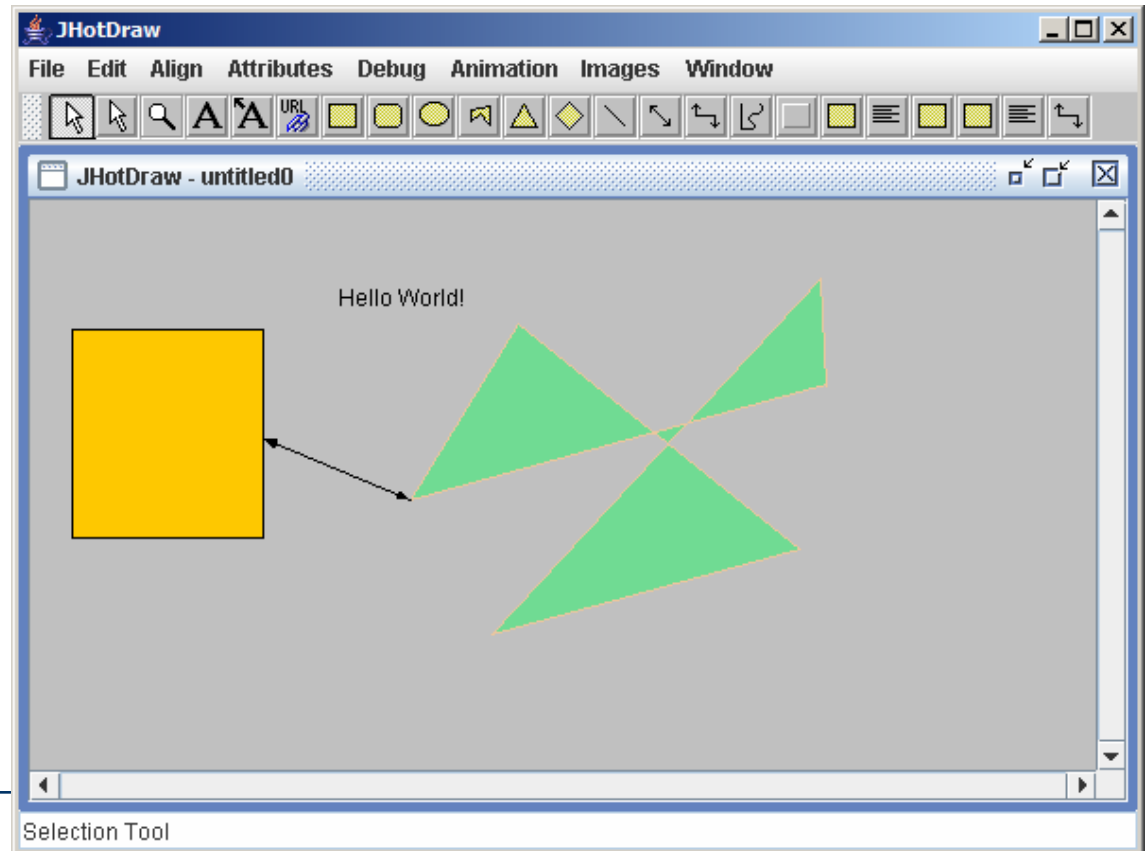
- Navigation to code for free
- High- and low-level comprehension side-by-side
- Documentation can rely on code for scaffolding
- Easier to keep up to date?

- Disadvantages

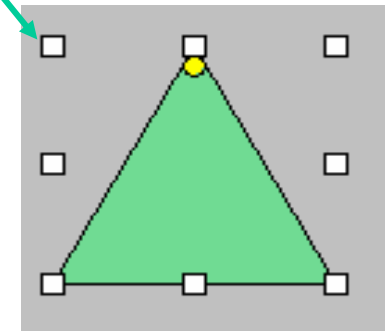
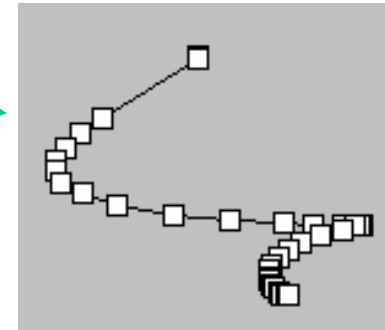
- Maintenance of documentation is delocalized
- Dependency on tool
- High- and low-level documentation harder to tell apart?

- Perfect setting for a controlled experiment.
  - P = Plain Text
  - J = JTourBus
- Software: JHotDraw
- Task: Corrective maintenance
- Subjects: Software engineering students
- Documentation process: Two stage
- Dependent Variables:
  - Time to solve task
  - Correctness of proposed solution

- Graphical editor framework and several applications by Gamma & Eggenschwiler
- Full of design patterns (448 classes)
- Well documented: 34KLOC comments, 26 KLOC code, UML diagrams



- Task posed to subjects:
  - ScribbleTool draws scribbles
  - No resizing possible as with triangle resize handles
  - Which code locations need to be modified to implement resizing?
- Instructions to subjects
  - **Understand** the system **sufficiently** to **generate a hypothesis** about the question
  - No need to validate it or write any code
- 90 Minutes



- 59 volunteer software engineering students
  - third and fourth year
- A similar task with different software was featured in a previous exercise
- Random assignment
  - Participants did not know there were two groups
- Both groups received
  - general information about the task,
  - a broad introduction to JHotDraw
- Subjects using JTourBus received an introduction to the tool

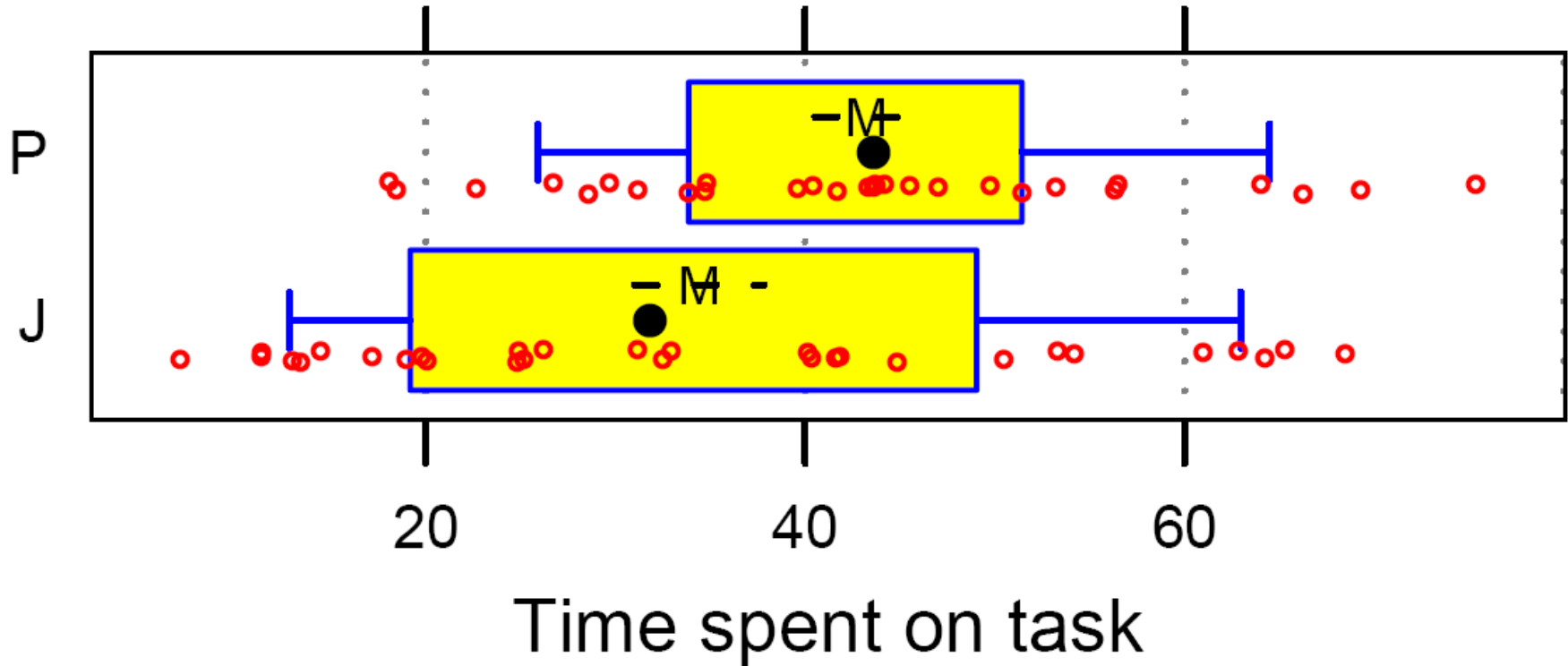
- Goals:
  - Same information present in P and J
  - High quality
  - No bias towards J
- Use top students from previous SE courses
  1. Pair of students produce J and P documentation for concern related to task together.
  2. Use third student to make sure both documents contain the same correct information
- Result:
  - 99 lines (P)
  - 107 lines in 15 stops, 2 tours (J)

	J	P
Correct	8	8
Incorrect	13	16
No answer	9	5

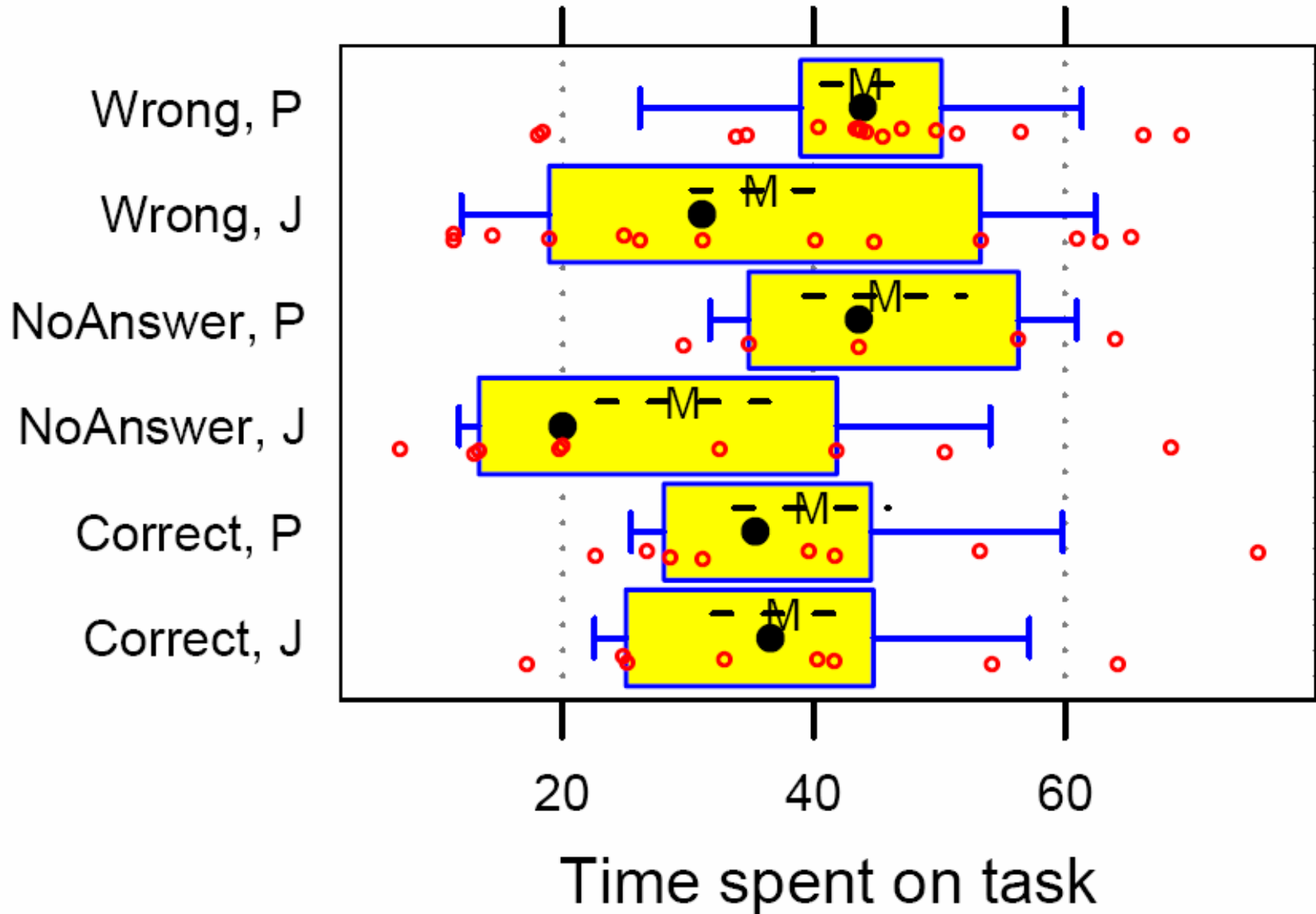
⇒ No difference in correctness of answers between J and P

- Question was very difficult
- 76% of the subjects formulated a hypothesis

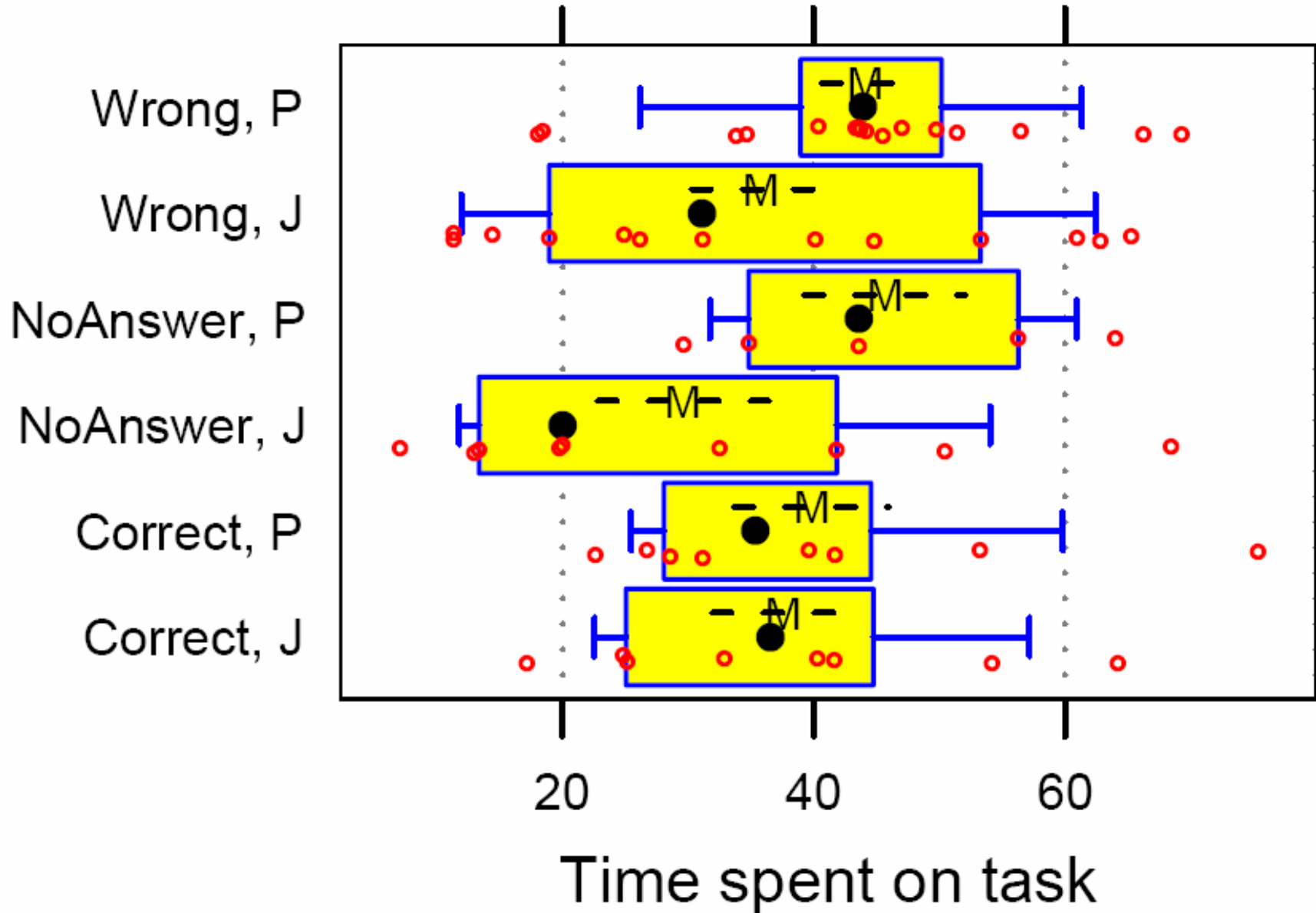
- Who answered correctly?
  - Neither Java experience nor IDE knowledge correlated with correct answers.
  - 88% of all correct answers by subjects with more than 40 points in SE exam (top 2/5th)
  - 81% of all incorrect and no-answers by subjects with 40 or less points



- JTourBus 9 minutes faster (23% of average time)



- No Answer:
  - JTourBus eliminates 2nd Order Ignorance (i.e. produces awareness about the things one does not know)
    - Thus if the task is too complex, one realizes it faster
  - Higher number of „No answers“ supports this
- Correct and incorrect answers:
  - Hypothesis 1: Being close to code increases the information in-take and leads to the need to pose a hypothesis.
  - Hypothesis 2: Tour-style documentation frustrates faster.
    - The format? The delocalization?  
The closer presence of the code?
  - Hypothesis 3: ???



**Thank you!** 