This is the coding scheme guide for identifying actions, task types, and biases. There are 4 steps in the coding process.

Step 1: Codifying the Transcripts

Look at the Screen Capture videos for each participant and transcribe in the following fields in the given format:
Description - a description of what they are doing, within which file or application. Be as detailed as possible.
Quote - What the participant was saying while performing the actions
Action Code - Will be completed in **Step 2**
Bias Categories - Will be completed in **Step 3**

Example Transcription

Index	Time stamp	Description	Quote	Action Code	Episode	Subgoal
1	hh:mm:ss	looking for a code file xx within package explorer	"I want to open this package explorer here so I can figure out where I want to put this in"			
2	hh:mm:ss	Pasted ‘defn-get-model’ from earlier copy into equ.symbolic-executer.set-caller	“I have to get a copy of both the ‘officiality’ and the ‘model’ but this is some non-versatile. I’ll put it here now.”			

Step 2: Codifying the Actions

Update the Action Codes in the Transcription Document.

Use the following table to identify the action for each row in the transcription document.
***Note*: Each (transcription) row might have more than one action associated.

Action Code Definitions

Codes	Definitions
Read	Examining information from artifacts (e.g. code, documentation, terminal output)
Edit	Any change made directly to code or related artifacts
Navigate	Moving within or among artifacts (e.g. pulling files from Git, opening files, scrolling through a file)
Execute	Compiling and/or running code
Ideate	Constructing a mental model of future changes

Action Code Rulebook

Here are some examples of the action codes:

Actions / Intentions	Instances	Positive Examples	Negative example
read	Reading code, documents, etc. Reading error messages or output. Examining VCS output.	Reading "error: extra-var-decls" Read stack overflow answers	Copies 'defn-collect-all-substring-ast' and 'defn- primitive-type' Move to package explorer window to find a file
edit	Permanently changing the code. Experimental or temporary changes to code. Staging or committing to the remote repository.	modifies the [x] attribute by adding % classList property value "xx" and deleted it added a console log statement	Search for 'this.class' within file xx.ts Print variable value directly from terminal
navigate	Move to a different window outside the IDE. Move to a different file within the IDE (through package explorer or hotkeys) Move to a different application. Launch a build/environment (ex. Loading REPL)	Clicked on the browser window in the background Closes out Terminal window Search for 'this.class' within file xx.ts	Scroll through the output or code.
execute	Running code to validate models Pushing to VCS repository	Types python run.py, presses enter. Starting nREPL server from file X	Added a console log statement
ideate	Constructing mental models of future changes, locating a bug, reason about behavior. Hypothesizing	"To start, I'm going to take a look at what this component [x] is so that I can more easily find it in the IDE." "Alright I am going to check., what I have done"	"I'm using the console to see if the data is showing up correctly"

Step 3: Biases

Cognitive Biases: Systematic patterns of deviation from optimal reasoning.

Use the following table to identify the cognitive bias category of each action:
***Note*: Each row in the transcription can more than one 'Category' from the following table.

Cognitive Bias Definitions

Category	Bias(es)	Coding Rules	Effects	Consequence
Preconception	Confirmation, Selective Perception	Decides which solution to work with based on -preconception/mental models -most familiar and available in memory [recall] -easiest to work and temporary fix E.g. "The problem takes x, okay I'll use hashmap;" "The easiest thing I can do now is z"	Memory/knowledge, narrowed vision	Inadequate Exploration
Ownership	IKEA effect, Endowment effect	Choose/create self authored method/document/element. E.g. "I'll use my own code because of x"	Solution space shrunk	Inadequate Exploration
Fixation	Anchoring and adjustment, Belief preservation, Semmelweis reflex, Fixation	Fixated on an element/method/code and continuously fixes it OR fixated on a cause and believes that is the reason. E.g. Repeated edits on a method call	Awareness about overall task	Preserving Context, Misplaced Attention
Resort to Default	Default, Status-quo, Sunk cost	Uses an existing solution that is preselected in a list OR that is already available.	Solution space shrunk	Inadequate Exploration, Misplaced Attention

		E.g. "This code uses a list, I'll just use that."		
Optimism	Valence effect, Invincibility, Wishful thinking, Overoptimism, Overconfidence	Optimistic that changes will work without things going wrong. E.g. "If I change x, the problem will be solved."	Decision too fast	Inadequate Exploration, Reduced Sense-making, Preserving Context
Convenience	Hyperbolic discounting, Time-based bias, Miserly information processing, Representativeness	Believe that there is a simple cause for every problem. Ignore information that contradicts belief. E.g. "Oh, it's just x."	Decision too fast	Inadequate Exploration, Reduced Sense-making
Subconscious action	Misleading information, Validity effect	Acts based on information provided by IDE without thinking OR based on information that is most frequently provided. E.g. chooses to change variable based solely on debugger suggestion. *Keep an eye out for information that appears repeatedly.	Not understanding the situation correctly	Reduced Sense-making
Bliss ignorance	Normalcy effect	Thinks that everything is normal even IDE/System/Server etc. failed.	Not paying attention to situation - overall situation	Reduced Sense-making, Preserving Context, Misplaced Attention
Superficial Selection	Contrast effect, Framing effect, Halo effect	Choose solution/element/method etc. based on superficial qualities. -especially in presence of contrasting elements -whether the element is presented in	The decision is not based on functionality	Reduced Sense-making, Misplaced Attention

		+ve/-ve light -assumes certain qualities based on known qualities E.g. “This StackOverflow post seems more recent, thus it must be more relevant.”		
Memory Bias	Primacy and recency, Availability	Recalls the first/the most recent of a list of alternatives. Keep an eye out for when participants encounter a list of alternatives, but uses only the last few/the first element	Memory affects decision	Inadequate Exploration

Unobserved Biases

Use the following table to identify the seven cognitive biases that we were unable to observe

Bias	Definition
Attentional Bias	The tendency of our perception to be affected by our recurring thoughts
Bandwagon Effect	The tendency for align stated opinions with the perceived majority opinion
Hindsight Bias	The tendency to pretend to have predicted the outcome all along after learning the actual outcome
Impact Bias	The tendency to overestimate the length or the intensity of future feeling states
Information Bias	The tendency to request unnecessary or unhelpful information, especially in times of uncertainty
Infrastructure Bias	The tendency for future economic and social development to be influenced by pre existing infrastructure such as roads
Mere Exposure Effect	The tendency to have an increased liking for stimulus due to continued exposure
Neglect of Probability	The tendency to disregard probability during decision-making
Semantic Fallacy	The tendency to pay less attention to the semantics of model than to its syntax or structural constraints

It is likely we did not witness these seven biases because of our study setup. We observed developers individually during their programming session; this limited the possibility of observing social interactions, which can lead to *bandwagon effect*. Our observation sessions were approximately 45 minutes long, which prevented us from observing biases with longitudinal effects, such as *mere exposure effect*, *impact bias*, *hindsight bias*, or *attentional bias*. Furthermore, there was no opportunity to observe *infrastructure bias*, as it is a systematic concern that affects the entire organization. Finally, we could not codify *information bias* since it was not possible to discern between information that a participant considered necessary or unnecessary in-situ.

Of the 30 biases that could be observed during software development tasks, we did not observe the following: *neglect of probability* and *semantic fallacy*. We did not find instances of *neglect of probability bias* as participants primarily focused on developing software, whereas, this bias is more likely to occur during planning or design sessions. We could not identify *semantic fallacy* due to the ambiguities of understanding when participants were actively focusing on syntax, and when that attention was appropriate and necessary.