Informed Consent

Please read the following informed consent form carefully before participating in this validation survey.

The purpose of this research study is to observe how developers create, manage, and recall contextual information when they are working on their "everyday" development task. Understanding the process will inform future tool development in creating intelligent and contextualized support for developers.

Results of this study may be used for fulfilling student dissertation and academic publications. This project is funded by National Science Foundation (NSF).

Activities: The study previously included observing you working on our regular day-to-day tasks. We are now conducting follow-up validation surveys to confirm our observations and results from that study. You will be asked to read and respond to a series of questions regarding your organization of programming tasks.

Study duration: The validation survey is expected to take 10-15 minutes to complete.

Storage and Future use of data: Data will be stored on protected drive on the researcher's computer and uploaded to a password-protected storage service offered/provisioned/procured by <<anonymized field>>. Access to data will be restricted to the researchers listed on the research protocol. Any data we collect will be kept confidential adhering to the guidelines of Institutional Review Board (a committee that reviews and approves research studies involving human subjects) and the agreements we make with your organization. All data will be anonymized as soon as possible.

Risks: There are no foreseeable risks to participating.

Benefits: There are no direct benefits for participating in the study. However, the study may influence the intelligence of future development support tools.

Compensation: There is no monetary compensation for participating in the study

Participation: Your participation in this study is voluntary. You are free to withdraw at any time. If you choose to withdraw from the study before it ends, your data from the session we will not be used. Your participation or non-participation will not have an impact on your relationship with <<anonymized field>> or your job.
Questions If you have any questions about this research project, please contact <<anonymized field>>, at <<anonymized field>> or by email at <<anonymized field>>. If you have questions about your rights or welfare as a participant, please contact the <<anonymized field>>, at <<anonymized field>> or by email at <<anonymized field>>.

Your acceptance indicates that this study has been explained to you, that your questions have been answered, and that you agree to take part in this study.

I understand and consent to participate in this study.

- ○ Yes
- ○ No

Information Abstraction

For the following four questions, please read the description of a task that Charlie, a fictional software developer, needs to complete. To complete the task, Charlie had to take multiple steps that are listed vertically on the left side. These **task steps** are ordered based on time.

We have also listed sets of **information elements** that may be relevant to the task at the top.

For each task step, select ALL information elements that you consider relevant to complete the step. An information element can be relevant to multiple task steps OR none.

Task Description: Charlie needs to configure the development environment prior to coding.

	REPL error message	Github Comment for CIDER version	Email discussing project status meeting schedule	Code snippet from email containing query structure	Query structure in file `query_interface.clj`
Charlie launches a REPL instance in a terminal window.	☐	☐	☐	☐	☐
Charlie locates an email from a colleague which describes Clojure query statements.	☐	☐	☐	☐	☐
The REPL instance returns an error.	☐	☐	☐	☐	☐
Charlie updates the CIDER version (Clojure Interactive Development Environment that Rocks).	☐	☐	☐	☐	☐
Charlie launches a REPL instance in a terminal window.	☐	☐	☐	☐	☐
Charlie opens the email from a colleague which describes Clojure query statements.	☐	☐	☐	☐	☐
Charlie locates a Clojure file that contains a query statement.	☐	☐	☐	☐	☐

Task Description: Charlie needs to implement a new experimental MonkeySort algorithm in Java.

	Syntax highlights indicating unknown variable name	Java compile error message indicating reference error	Java compile error message indicating memory out of bounds	Java data structures for storing incoming data	Line count for total size of Java file
Charlie implements a naive initial version of the algorithm.	☐	☐	☐	☐	☐
Charlie executes the code in a terminal window.	☐	☐	☐	☐	☐
The terminal returns a reference error for a unknown variable name.	☐	☐	☐	☐	☐
Charlie updates the variable names in the algorithm code.	☐	☐	☐	☐	☐
Charlie executes the code in a terminal window.	☐	☐	☐	☐	☐
The terminal returns a memory overflow error.	☐	☐	☐	☐	☐
Charlie updates the storage portion of the algorithm code.	☐	☐	☐	☐	☐
Charlie executes the code in a terminal window.	☐	☐	☐	☐	☐

Task Description: Charlie needs to review code from a colleague before integrating it.

	Editor inline documentation box for hashSort	Editor inline documentation box for setPivot	External documentation for sortAll	Command line output from running test suite	Blog post describing reverseSort algorithm
Charlie examines a new sortAll command that uses a command called hashSort.	☐	☐	☐	☐	☐
Charlie opens the definition of the hashSort command.	☐	☐	☐	☐	☐
Charlie examines the definition of hashSort and finds a setPivot command.	☐	☐	☐	☐	☐
Charlie opens the definition of the setPivot command.	☐	☐	☐	☐	☐
Charlie examines the definition of the setPivot command.	☐	☐	☐	☐	☐
Charlie returns to the definition of hashSort and finishes examining the definition.	☐	☐	☐	☐	☐
Charlie returns to the definition of sortAll and finishes examining the definition.	☐	☐	☐	☐	☐

Task Description: Charlie needs to debug code which is throwing errors after adding a new function.

	Error message indicating unknown `hover_node` variable	Error message indicating memory overflow	Error message indicating no `id` value on `select_node` element	Code definitions within `hierarchyService.ts`	Code definitions within `interactionService.ts`
Charlie creates a `hover_node` variable in `hierarchyService.ts`.	☐	☐	☐	☐	☐
Charlie compiles and executes the project.	☐	☐	☐	☐	☐
Charlie adds an import for a storage library in `hierarchyService.ts`.	☐	☐	☐	☐	☐
Charlie compiles and executes the project.	☐	☐	☐	☐	☐
Charlie copies an import library from `interactionService.ts` and pastes into `hierarchyService.ts`.	☐	☐	☐	☐	☐
Charlie compiles and executes the project.	☐	☐	☐	☐	☐
Charlie writes a new method to set the `hierarchyID` variable.	☐	☐	☐	☐	☐

Task Description: Charlie needs to implement the output methods for working with a hashmap data structure.

	Clojure API documentation for `some()`	Clojure API documentation for `contains()`	Command line output	Clojure API documentation for `has()`	Code definitions for output method using `contains()`
Charlie creates an output method that uses the `some()` function.	☐	☐	☐	☐	☐
Charlie compiles and executes the project.	☐	☐	☐	☐	☐
Charlies creates a different version of the output method that uses the `contains()` function.	☐	☐	☐	☐	☐
Charlie removes redundant looping code from the output method that uses `some()`.	☐	☐	☐	☐	☐
Charlie compiles and executes the project.	☐	☐	☐	☐	☐
Charlie removes redundant looping code from the output method that uses `contains()`.	☐	☐	☐	☐	☐
Charlie copies a portion of code from the output method that uses `some()` and removes the method.	☐	☐	☐	☐	☐
Charlie pastes into the output method that uses `contains()`.	☐	☐	☐	☐	☐

Patterns in Task Structuring

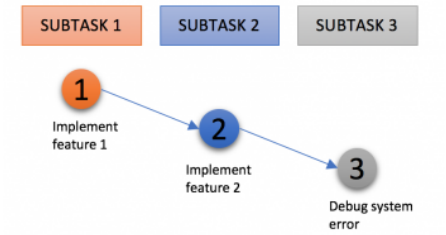
For the following five (5) questions, please read the descriptions of the patterns in which developers tend to organize their programming tasks. For each pattern, please select the option that most closely represents the frequency that you use the pattern.

After learning about the five patterns, please rank them in order of *most* to *least* frequently used patterns in your normal tasks. Based on your ranked responses, we will ask you to describe when and why you use your most frequent and least frequent patterns.

Please read the descriptions of the patterns that Charlie uses to organize his programming tasks.

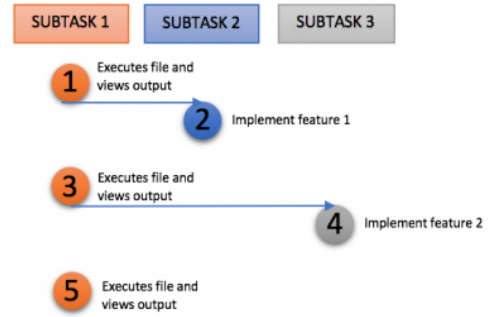
For each pattern, please select the option that most closely represents how frequently you use the pattern.

Sequential Pattern: A sequence of subtasks. Charlie, completes one subtask to move on to the next subtask.



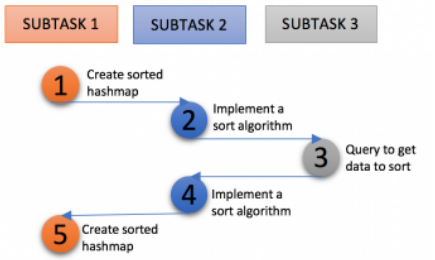
Never	Rarely	Sometimes	Most of the time
○	○	○	○

Grounding Pattern: A return to a previous subtask to evaluate progress. After writing code for a bug fix, Charlie evaluates whether the new code works correctly in the larger program.



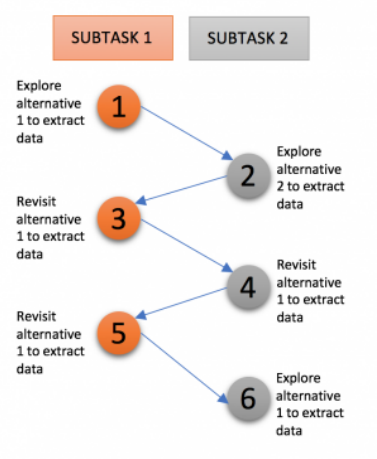
Never	Rarely	Sometimes	Most the time
○	○	○	○

Recursion Pattern: Tasks are tackled in nested steps. Charlie needs to complete the smaller subtasks to continue with the larger.



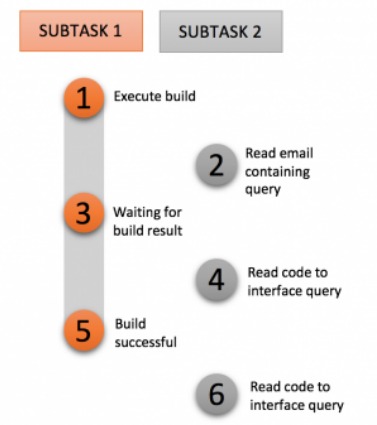
Never	Rarely	Sometimes	Most of the time
○	○	○	○

Alternating Pattern: Working on two subtasks simultaneously, where both subtasks are alternate solutions to the same problem.

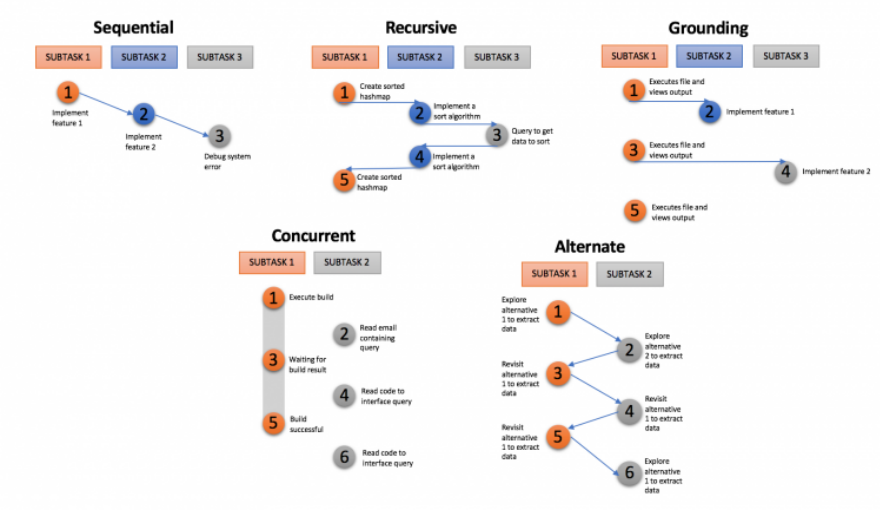


Never	Rarely	Sometimes	Most of the time
○	○	○	○

Concurrent Pattern: Working on two subtasks at the same time. Charlie offloads one subtask as a running process. While waiting for that process to complete he works on another subtask.



Never	Rarely	Sometimes	Most of the time
○	○	○	○



Rank the following patterns by dragging them up and down based on the frequency in which you use each pattern. (1 - most frequent, 5 - least frequent)

- Alternating
- Sequential
- Grounding
- Recursion
- Concurrent

For your most frequent pattern, **${q://QID1/ChoiceGroup/ChoiceWithLowestValue}**, **when** do you use this pattern?

For your most frequent pattern, **${q://QID1/ChoiceGroup/ChoiceWithLowestValue}**, **why** do you use this pattern?

For your least frequent pattern, **${q://QID1/ChoiceGroup/ChoiceWithHighestValue}**, **when** do you use this pattern? If never, please indicate when you would possibly use this pattern.

For your least frequent pattern, **${q://QID1/ChoiceGroup/ChoiceWithHighestValue}**, **why** do you use this pattern? If never, please indicate why you do not use this pattern.