A Qualitative Study on the Implementation Design Decisions of Developers

Check out our paper!

Jenny T. Liang[1], **Maryam Arab**[2], **Minhyuk Ko**[3], **Amy J. Ko**[4], **Thomas D. LaToza**[2]
Carnegie Mellon University[1], *George Mason University*[2], *Virginia Tech*[3], *Univeristy of Washington*[4]

Background

Motivation

Decision-making is a key software engineering skill. Developers constantly make choices throughout the software development process, from requirements to implementation.

While prior work has studied developer decision-making, **the choices made while choosing what solution to write in code** is understudied.

Study Design

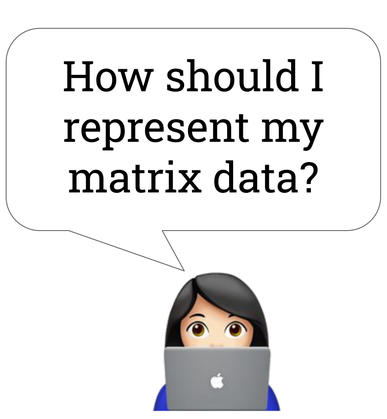
A **Google Forms survey** (n = 46) collecting recent **implementation decisions** and **considerations** in the decision-making process.

Semi-structured interviews (n = 14) on collecting recent **implementation decisions**, **considerations** in the decision-making process, and **strategies** used to make decisions.

Implementation Design Decisions

Definition: When developers select one specific way to implement a behavior, given many potential alternatives.

Example: Choosing to handle matrix data by using native Python arrays, instead of C or Numpy, in order to optimize for readability over performance (e.g., runtime, memory).



Decision Types (9 total)

Decision Type	Description
Behaviors	Deciding the program specification (e.g., parameters or returns of a method).
Code constructs	Deciding which programming language constructs to use within a program.
Structure	Deciding how to organize the codebase, where files should lie, and how code should be modularized.
Languages, APIs, services	Deciding the programming languages, APIs, or third-party services to use in the software system or script.
Automation	Deciding whether to implement a technology solution from scratch.

Considerations (25 total)

Code	Description
Community support	How well-supported by a developer community a technology is.
Consistency	Being consistent with the code style of the programming language or code base
Impacts	The impacts that the implementation may cause
Future requirements	Requirements or customer needs that may or may not occur in the future.
Maintainability	How easily maintenance actions (e.g., fixing defects, updating components) can be performed on software
System fit	How well the implementation fits in with an existing code base or system.
Requirements	The requirements of the software; customer needs.
Reliability	How reliable and correct the software is
Reusing resources	Reusing existing resources (e.g., code, practices).

Decision-Making Processes

Example participant data

Use this when: Using less common features in libraries instead of using the popular functions
Tools/technologies: StackOverflow, Google, continuous learning
Prior knowledge: Common design patterns, popular libraries

1) Decide what the goal of the program is.
2) Begin writing the program.
3) While writing the program, search online whether other libraries support your use case...
4) Choose a library which meets your use case....
5) Look at the features of the library and test the ones that you're interested in on small examples. Get a feel of the library and select a solution which achieves the desired behavior.
6) If you have code that works, show the solution to another individual for review.

Action	Median Position
Providing context	1.5
Researching	2
Defining requirements	2
Brainstorming	3
Estimating	3
Evaluating	4
Proof-of-concept	6.5
Updating requirements	7
Implementing	7
Reviewing	8
Testing	9
Updating implementation	9
Deploying	10.5

Implications

Takeaway 1: Implementation design decisions **are shaped by higher levels of design**, and vice versa.

Takeaway 2: Making implementation design decisions **is both an art and a science**.

Carnegie Mellon University · George Mason University · Virginia Tech · University of Washington

ICSE2023