Prompts Are Programs Too!
Understanding How Developers Build Software Containing Prompts

Jenny Liang et al.

> "I suspect that machines to be programmed in our native tongues... are as damned difficult to make as they would be to use."
>
> — Edsger W. Dijkstra (1979)

Context

Prompt-powered software like MS Copilot, Google AI Search, now agentic AI assistants are used by millions of people

Gap

The literature studies "prompt engineering" in limited contexts

This Paper

Defines and studies previously unidentified "prompt programming": prompts authored at design time but executed at runtime on user input

Research Question:

How do programmers develop programs that incorporate natural language prompts?

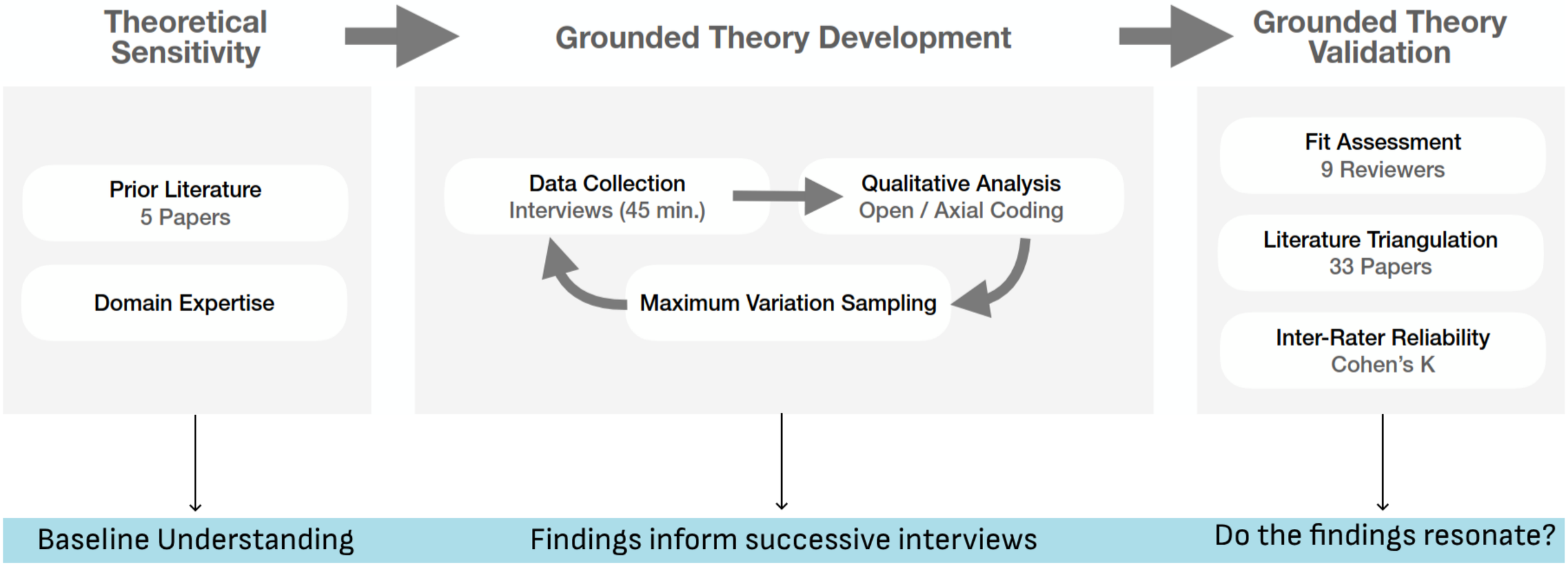
RQ Type:

- Broad and Exploratory
- Based on limited prior research

Method:

- Straussian Grounded Theory
- Systematic method to develop a theory from data

Methodology: Straussian Grounded Theory Process



Interview Protocol

Setup

- 45 min Zoom Interviews
- $20 compensation
- IRB Approved
- 17/20 showed prompts

Interview

1. Background survey
2. Define prompt program
3. Recall last prompt program
4. Discuss challenges
5. Discuss dev process

Analysis

- 3 authors
- Independent coding
- Merged codebooks
- 15 key observations
- Grouped by vote into 4 barriers

"Did your prompt change after deployment?"

"Did you use metrics to determine when the prompt was successful?"

"How did you identify what caused an incorrect output?"

Who Did They Interview?

Number of Participants	20
Experience	3–20 years, median 8.5
Prompt Programs Written	1–100+, median 10
Domains	security, education, robotics, etc.
Contexts	academia, big tech, startups, freelance, OSS
Prompt Types	single prompts, multi prompts, agents
Models Used	open & closed source, LLMs, VLMS

Maximum variation sampling, very diverse set of participants
Recruited via snowball sampling and GitHub

How Did They Interview?

Interview Style

· Semi-structured
· Retrospective
· Discussed challenges before process

Themes Explored

· Requirements
· Design
· Implementation
· Debugging
· Data curation
· Evaluation
· Deployment

Sample Questions

Think of the most recent time you wrote such a prompt. Do you have one in mind?

Do you currently have access to the prompt?

Did you experience any difficulties while using existing tools to develop a prompt?

How did you identify what caused an incorrect output? Were you confident in your answer?

Threats to Validity

Internal Validity

- Memory biases
  - Asked participants to review their prompts
- Confirmation bias from authors
  - Could affect categories and final theory
  - Mitigated by lit triangulation? (?)

External Validity

- May not generalize
- Recruiting from author social networks
- Self-selection bias
- Secretive company policy
- Quickly evolving tech

Conclusion Validity

- Qualitative, relies on interpretation
- Mitigations:
  - Authors attended multiple interviews
  - Theory developed iteratively with unanimous agreement
  - Assessed inter-rater reliability
  - Validated with eight participants and an expert

Summary

RQ:

"How do programmers develop programs that incorporate natural language prompts?"

· Broad, exploratory RQ to develop grounded theory from limited prior knowledge

Straussian Grounded Theory

· Literature triangulation
· Interviews
· Fit check

Interview Methods

· Maximum variation sampling
· Artifacts presented
· Asked challenges before process
· Continued after theoretical saturation

Findings (4 barriers)

· Understanding FM behavior (unreliable mental models)
· Accomodating for randomness (prompts are "finicky" or "fragile")
· Programming in natural language (modularization and explicitness)
· Testing prompt programs (qualitative success, requires representative data)