How to Write Commit Messages

Applied Research Using the DOT Framework

1. Research Overview

Research Question

How can developers write clear, well-structured commit messages that improve team communication and project maintainability?

This question aims for a practical, actionable outcome rather than theoretical insight alone — perfect for DOT-style research.

ICT Research Methodologyi

2. Research Design According to DOT

The Development-Oriented Triangulation (DOT) framework organizes research into three aspects:

- What? (Different domains of knowledge)
- Why? (Justification and trade-offs)
- How? (Research strategies and methods)

ICT Research Methodologyi

We focus on:

- Application domain: commit messages in software development
- Available work: existing guidelines, conventions, best practices
- Innovation domain: synthesizing guidelines into a clear practical method

Why these methods?

To understand how to write commit messages, we combine methods that provide:

- theoretical foundation (Library methods)
- real-world interaction and patterns (Field methods)
- structured task understanding (Field task analysis)

Using more than one method improves confidence (triangulation). ICT Research Methods

3. Research Methods

We apply the DOT framework by selecting multiple complementary methods:

A. Literature Study (Library)

Purpose:

To collect existing guidelines, best practices, and rules on commit messages from reliable sources.

Why?

Literature studies find general information and established norms that help define how practice should look.

Literature study

Activities:

- Identify authoritative sources on commit messages
- Extract recommendations and principles
- Compare and analyse differences

Why this fits:
Literature study is suitable for gathering expert knowledge and standards in the field.

Literature study

B. Document Analysis (Field)

Purpose:

To analyse real commit history examples to see how developers actually compose commit messages.
Why?
Document analysis helps understand current practices and gaps between guidelines and reality.

Document analysis

Activities:

- Collect commit logs from open-source projects
- Categorize message formats (clear, ambiguous, missing context)
- Identify patterns and common mistakes

Why this fits:
Document analysis is effective for observing behaviour without interference.

Document analysis

C. Task Analysis (Field)

Purpose:
To understand the actual task of writing a commit message: what steps are involved, what decisions developers make.
Why?
Task analysis focuses on how users perform a task and what challenges arise.

Task analysis

Activities:

- Decompose the writing process (read code, decide intent, format message)
- Map common decision points
- Identify cognitive load or ambiguities in steps

Why this fits:
Task analysis improves understanding of practical behaviour, not just what should be done.

Task analysis

4. Results & Findings

A. From Literature Study

Sources show high-level recommendations for commit message structure:

- Short, descriptive summary (≤ 50 characters)
- Optional body explaining why the change was made and context
- Consistent style improves readability and tooling support

Guidelines such as the Git documentation and conventions confirm this structure.

Git commit guidlines

These reflect theoretical best practices — they are widely cited in developer documentation and style guides.

B. From Document Analysis

Analysis of real commit logs reveals:

Common good practices:

- Clear subject lines with action verbs
- Short but descriptive messages
- References to issues or bug IDs

Common issues:

- Messages like "fix stuff" or "updates" (low clarity)
- Mixed changes in one commit message
- No explanation of why the change occurred

These patterns match research expectations about real behaviour vs. guidelines.

C. From Task Analysis

Breaking down the process of writing a commit message shows developers need to:

1. Identify change intent (What is the core purpose?)
2. Describe the modification succinctly
3. Add context if non-trivial
4. Follow team or project conventions

Challenges often occur at steps 1 & 3 — deciding why the change was made is cognitively heavier than summarizing code diffs.

5. Synthesised Practical Guidelines

Based on triangulated research findings, the following practical method for commit message writing is proposed:
Commit Message Template

1. Subject line (imperative mood):

- Format: [scope/context]
- Example: Fix login crash when token expired

2. Blank line

3. Body (optional but recommended for non-trivial changes):

- Explain why the change was necessary
- Include references (e.g., issue/Ticket ID)
- Provide clarity about decisions

Checklist Before Commit

✓ Does the subject line summarize the change clearly?
✓ Did you use imperative mood?
✓ If needed, did you add context and rationale?
✓ Is the message formatted with clear separation and no ambiguity?

6. Conclusion

Using the DOT framework, we combined:

- Literature Study to collect accepted norms
- Document Analysis & Task Analysis to understand real practice and steps.

This mixed approach revealed both best practices and common pitfalls, yielding a method that is:

- grounded in actual developer behaviour
- supported by authoritative sources
- applicable across teams and workflows