Wirasm / PRPs-agentic-eng

Code · Issues 6 · Pull requests 4 · Actions · Projects · Security · Insights

Prompts, workflows and more for agentic engineering

1.9k stars · 581 forks · 65 watching · Branches · Activity · Tags

Public repository

3 Branches · 0 Tags · Go to file · Go to file · Add file · Code

Wirasm refactor: make prp-core commands project-agnostic · 90419a0 · 3 days ago

.claude-plugin	feat: add prp-core plugin with mark...	3 months ago
.claude	refactor: make prp-core commands...	3 days ago
claude_md_files	Merge branch 'Wirasm:developmen...	6 months ago
old-prp-commands	chore: move PRPs folder to old-prp...	last week
plugins/prp-core	refactor: make prp-core commands...	3 days ago
.gitignore	chore: update project config and a...	3 months ago
.python-version	feat: initialize project with PRP tem...	7 months ago
CLAUDE.md	feat: add agent skill for prp core wo...	3 months ago
README-for-DUMMIES.md	feat: add /prp-debug command for...	last week
README.md	feat: add /prp-debug command for...	last week
pyproject.toml	chore: update project config and a...	3 months ago
uv.lock	chore: update project config and a...	3 months ago

PRP (Product Requirement Prompts)

A collection of prompts for AI-assisted development with Claude Code.

Video Walkthrough

https://www.youtube.com/watch?v=KVOZ9s1S9Gk&lc=UgzfwxvFjo6pKEyPo1R4AaABAg

Support This Work

Found value in these resources?

Buy me a coffee: https://coff.ee/wirasm

I spent a considerable amount of time creating these resources and prompts. If you find value in this project, please consider buying me a coffee to support my work.

Transform Your Team with AI Engineering Workshops

Ready to move beyond toy demos to production-ready AI systems?

Book a workshop: https://www.rasmuswiding.com/

What you'll get:

- Put your team on a path to become AI power users
- Learn the exact PRP methodology used by top engineering teams
- Hands-on training with Claude Code, PRPs, and real codebases
- From beginner to advanced AI engineering workshops for teams and individuals

Perfect for: Engineering teams, Product teams, and developers who want AI that actually works in production

Contact me directly at hello@rasmuswiding.com

What is PRP?

Product Requirement Prompt (PRP) = PRD + curated codebase intelligence + agent/runbook

The minimum viable packet an AI needs to ship production-ready code on the first pass.

A PRP supplies an AI coding agent with everything it needs to deliver a vertical slice of working software—no more, no less.

How PRP Differs from Traditional PRD

A traditional PRD clarifies *what* the product must do and *why* customers need it, but deliberately avoids *how* it will be built.

A PRP keeps the goal and justification sections of a PRD yet adds AI-critical layers:

- **Context**: Precise file paths, library versions, code snippet examples
- **Patterns**: Existing codebase conventions to follow
- **Validation**: Executable commands the AI can run to verify its work

---

Quick Start

Option 1: Copy Commands to Your Project

```
# From your project root
```

```
cp -r /path/to/PRPs-agentic-eng/.claude/commands/prp-core .claude/commands/
```

Option 2: Clone Repository

```
git clone https://github.com/Wirasm/PRPs-agentic-eng.git
cd PRPs-agentic-eng
```

Commands

The `.claude/commands/prp-core/` directory contains the core PRP workflow commands:

Core Workflow

Command	Description
`/prp-prd`	Interactive PRD generator with implementation phases
`/prp-plan`	Create implementation plan (from PRD or free-form input)
`/prp-implement`	Execute a plan with validation loops

Issue & Debug Workflow

Command	Description
`/prp-issue-investigate`	Analyze GitHub issue, create implementation plan
`/prp-issue-fix`	Execute fix from investigation artifact
`/prp-debug`	Deep root cause analysis with 5 Whys methodology

Git & Review

Command	Description
`/prp-commit`	Smart commit with natural language file targeting
`/prp-pr`	Create PR with template support
`/prp-review`	Comprehensive PR code review

Autonomous Loop

Command	Description
`/prp-ralph`	Start autonomous loop until all validations pass
`/prp-ralph-cancel`	Cancel active Ralph loop

Ralph Loop (Autonomous Execution)

Based on Geoffrey Huntley's Ralph Wiggum technique - a self-referential loop that keeps iterating until the job is actually done.

How It Works

```
/prp-ralph .claude/PRPs/plans/my-feature.plan.md --max-iterations 20
```

1. Claude implements the plan tasks
2. Runs all validation commands (type-check, lint, tests, build)
3. If any validation fails → fixes and re-validates
4. Loop continues until ALL validations pass
5. Outputs `<promise>COMPLETE</promise>` and exits

Each iteration, Claude sees its previous work in files and git history. It's not starting fresh - it's debugging itself.

Setup

The stop hook must be configured in `.claude/settings.local.json`:

```
{
  "hooks": {
    "Stop": [
      {
        "hooks": [
          {
            "type": "command",
            "command": ".claude/hooks/prp-ralph-stop.sh"
          }
        ]
      }
    ]
  }
}
```

Usage

```
# Create a plan
/prp-plan "add user authentication with JWT"
```

README

```
/prp-ralph .claude/PRPs/plans/add-user-auth.plan.md --max-iterations 20

# Cancel if needed
/prp-ralph-cancel
```

Tips

- Always use `--max-iterations` (default: 20) to prevent infinite loops
- Works best with plans that have clear, testable validation commands
- State is tracked in `.claude/prp-ralph.state.md`

- Progress and learnings are captured in the implementation report

---

Workflow Overview

Large Features: PRD → Plan → Implement

```
/prp-prd "user authentication system"
    ↓
Creates PRD with Implementation Phases table
    ↓
/prp-plan .claude/PRPs/prds/user-auth.prd.md
    ↓
Auto-selects next pending phase, creates plan
    ↓
/prp-implement .claude/PRPs/plans/user-auth-phase-1.plan.md
    ↓
Executes plan, updates PRD progress, archives plan
    ↓
Repeat /prp-plan for next phase
```

Medium Features: Direct to Plan

```
/prp-plan "add pagination to the API"
    ↓
Creates implementation plan from description
    ↓
/prp-implement .claude/PRPs/plans/add-pagination.plan.md
```

Bug Fixes: Issue Workflow

```
/prp-issue-investigate 123
    ↓
Analyzes issue, creates investigation artifact
    ↓
/prp-issue-fix 123
    ↓
Implements fix, creates PR
```

Artifacts Structure

All artifacts are stored in `.claude/PRPs/`:

```
.claude/PRPs/
├── prds/              # Product requirement documents
├── plans/             # Implementation plans
│   └── completed/     # Archived completed plans
├── reports/           # Implementation reports
├── issues/            # Issue investigation artifacts
│   └── completed/     # Archived completed investigations
└── reviews/           # PR review reports
```

PRD Phases

PRDs include an Implementation Phases table for tracking progress:

```
| #   | Phase | Description | Status      | Parallel | Depends | PRP Plan |
| --- | ----- | ----------- | ----------- | -------- | ------- | -------- |
| 1   | Auth  | User login  | complete    | -        | -       | [link]   |
| 2   | API   | Endpoints   | in-progress | -        | 1       | [link]   |
| 3   | UI    | Frontend    | pending     | with 4   | 2       | -        |
| 4   | Tests | Test suite  | pending     | with 3   | 2       | -        |
```

- **Status**: `pending` → `in-progress` → `complete`
- **Parallel**: Phases that can run concurrently (in separate worktrees)
- **Depends**: Phases that must complete first

PRP Best Practices

1. **Context is King**: Include ALL necessary documentation, examples, and caveats
2. **Validation Loops**: Provide executable tests/lints the AI can run and fix
3. **Information Dense**: Use keywords and patterns from the codebase
4. **Bounded Scope**: Each plan should be completable by an AI in one loop

Project Structure

```
your-project/
├── .claude/
│   ├── commands/prp-core/   # PRP commands
│   ├── PRPs/                # Generated artifacts
│   └── agents/              # Custom subagents
├── PRPs/
│   ├── templates/           # PRP templates
│   └── ai_docs/             # Library documentation
├── CLAUDE.md                # Project-specific guidelines
└── src/                     # Your source code
```

Parallel Development with Worktrees

When PRD phases can run in parallel:

```
# Phase 3 and 4 can run concurrently
git worktree add -b phase-3-ui ../project-phase-3
git worktree add -b phase-4-tests ../project-phase-4

# Run Claude in each
cd ../project-phase-3 && claude
cd ../project-phase-4 && claude
```

Resources

Templates (PRPs/templates/)

- `prp_base.md` - Comprehensive PRP template
- `prp_story_task.md` - Story/task template
- `prp_planning.md` - Planning template

AI Documentation (PRPs/ai_docs/)

Curated documentation for Claude Code context injection.

Legacy Commands

Previous command versions are preserved in `old-prp-commands/` for reference.

License

MIT License

Support

I spent a considerable amount of time creating these resources and prompts. If you find value in this project, please consider buying me a coffee to support my work.

Buy me a coffee: https://coff.ee/wirasm

Releases

No releases published

Packages

No packages published

Contributors 4

Wirasm Rasmus Widing

claude Claude

stavarc

eddiedunn Eddie Dunn

Languages

● **Python** 75.4% ● **Shell** 24.6%