# RDD Framework User Guide

Requirements-Driven Development Framework

A comprehensive guide for AI-assisted development with GitHub Copilot

# 📖 Table of Contents

# Introduction to RDD Framework

The Requirements-Driven Development (RDD) framework is a structured workflow system designed to streamline development with GitHub Copilot. It provides a simple, workspace-based approach to managing iterations, documenting requirements, and executing development tasks through AI assistance.

## What This Guide Covers

This comprehensive guide will teach you how to install and configure the RDD framework in your project, use the interactive terminal menu to manage development iterations, write and execute prompts for GitHub Copilot, maintain synchronized documentation throughout development, and follow best practices for effective AI-assisted development.

### Who This Guide Is For

This guide is written for intermediate developers comfortable with command-line interfaces, Git version control basics, VS Code or similar editors, and basic Python concepts.

# Installation Process

The RDD framework installation is straightforward and consistent across Windows, Linux, and macOS platforms. Before installing, ensure you have Python 3.7+ installed and accessible via the python command, Git 2.23+ for version control, and a Git repository initialized in your project directory.

## Download the Release

Visit the RDD GitHub Releases page and download the latest rdd-v{version}.zip file. Extract the archive to a temporary location on your system.

## Run the Installer

Navigate to the extracted directory and run the installer script. On Linux/macOS: `./install.sh`. On Windows: `install.bat`. You can also run directly with Python: `python install.py`

## Configure Installation

The installer guides you through options: choose your project folder, select local-only mode or GitHub remote, and specify your default branch (main, dev, master, or custom).

## Verify Installation

After completion, verify it's working by running `python .rdd/scripts/rdd.py --version` from your project directory. You should see: RDD Framework v{version} (Python)

**Note for Linux/macOS users:** The framework uses the python command. If not available, install python-is-python3 on Debian/Ubuntu (`sudo apt install python-is-python3`) or create an alias in your shell configuration.

# Initial Setup & Configuration

After installation, run the RDD menu to configure the framework for your project. You can start the menu using the launcher script (`./rdd.sh` on Linux/macOS or `rdd.bat` on Windows) or directly with Python: `python .rdd/scripts/rdd.py`

## Configuration Menu Options

Select option 5 from the main menu to access configuration where you can update version numbers (major, minor, or patch increments), change the default branch by selecting from existing branches, and toggle local-only mode to enable or disable GitHub remote operations.

Configuration changes are saved to `.rdd-docs/config.json` and persist across sessions. This ensures your preferences remain consistent throughout your development workflow.

```
╔═══════════════════════════════════════════════════════════╗
║ RDD Framework                                             ║
║ Requirements-Driven Development                           ║
╚═══════════════════════════════════════════════════════════╝

Current branch: main
Default branch: main

RDD Framework - Main Menu:
1. Create new iteration
2. Update from default
3. Complete current iteration
4. Delete merged branches
5. Configuration
9. Exit

Enter your choice (1-9):
```

**Tip:** When managing installations, check if you're in a virtual environment. It's generally better to let humans handle installations rather than letting Copilot install packages.

# Core Workflow Cycle

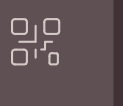
The RDD framework follows a simple, repeatable cycle for each development iteration. Understanding this workflow is essential to maximizing your productivity with AI-assisted development.

### Create New Iteration

Start from default branch with empty workspace. Provide branch name and framework creates branch, initializes workspace with prompts file.

### Define Work in Prompts File

Edit `.rdd-docs/work-iteration-prompts.md` and write clear, specific prompts for Copilot with full relative file paths.



### Execute Prompts with Copilot

Use `/rdd.execute` command in GitHub Copilot Chat. Copilot implements each prompt and creates implementation files in workspace.

### Update from Default (Optional)

Sync your branch with latest default branch changes and resolve any conflicts that arise.

### Complete Iteration

Archives workspace with timestamp, commits all changes, optionally pushes to remote, and returns to default branch.

### Create Pull Request

Use GitHub web interface to create PR and request code review from team members.

### Merge & Cleanup

After PR is merged, delete merged branches using menu option 4 to keep repository clean.

**Best Practice:** Start a new chat session for each prompt execution. This keeps the context focused and reduces confusion from previous conversations.

# Creating a New Iteration

Starting a new iteration creates a feature branch and sets up your workspace for development. Select option 1 from the main menu: **Create new iteration**.

## Prerequisites

- You must be on the default branch
- Workspace must be empty (`.rdd-docs/workspace/` should not exist or be empty)

If these conditions aren't met, the framework will display an error and guide you to fix the issue.

## Branch Name Entry

You'll be prompted to enter a descriptive branch name like "fix user login bug". The framework automatically normalizes it to kebab-case (fix-user-login-bug).

### Branch Creation

New Git branch is created and checked out automatically

### Workspace Initialization

Prompts file created from template at `.rdd-docs/work-iteration-prompts.md`


**Tip:** Don't write all prompts in advance. Execute one prompt, see the results, then add the next prompt. This iterative approach keeps you responsive to what Copilot discovers during implementation.

# Working with Prompts

The heart of RDD development is the `work-iteration-prompts.md` file where you define specific tasks for GitHub Copilot to execute. Located at `.rdd-docs/work-iteration-prompts.md`, this file structures your development work into discrete, executable units.

## Prompt File Structure & Components

```
# Work Iteration Prompts

## Prompt Definitions

- [ ] [P01] Create user authentication
module in `src/auth.py` with login
and logout functions. Include input
validation.

- [ ] [P02] Add unit tests for
authentication in `tests/test_auth.py`.
Cover valid login, invalid password,
and missing username cases.

- [x] [P03] Update README.md with
authentication usage examples
```

**Components breakdown:**

- **Checkbox:** `- [ ]` for pending, `- [x]` for completed
- **Prompt ID:** `[P01]`, `[P02]`, etc. (unique identifier)
- **Instructions:** Clear, detailed description with full file paths

> **Important:** Never manually edit checkboxes. Let the framework mark them automatically when prompts complete.

## Writing Effective Prompts

### √ Good Prompt Example

`[P01] Create a validation utility in `src/utils/validators.py` with functions for email validation (RFC 5322 format) and phone number validation (E.164 format). Include docstrings following Google style guide. Add comprehensive unit tests in `tests/utils/test_validators.py` with 100% coverage.`

Specific, actionable, includes paths, references standards, clear outcomes, single responsibility.

### ☒ Bad Prompt Example

`[P01] Add validation`

**Why it's bad:** No file path, no specifics on what to validate, no context or examples provided.

## Executing Prompts in Copilot Chat

Open VS Code with GitHub Copilot installed, launch Copilot Chat (Ctrl+Shift+I / Cmd+Shift+I), ensure you're using an appropriate model like Claude Sonnet 4 or GPT-4, then execute with `/rdd.execute P01` or just `/rdd.execute` to run the next uncompleted prompt automatically.

# Special-Purpose Prompts

The RDD framework includes two special prompts designed for specific phases of your development workflow: documentation synchronization and comprehensive requirement analysis.

## The .rdd.update Prompt

**Purpose:** Update documentation after completing development work

**Location:** `.github/prompts/rdd.update.prompt.md`

**When to Use:** After all development prompts are executed, before completing the iteration, to synchronize requirements and tech spec with code changes.

**Usage:** Execute `/rdd.update` in Copilot Chat or add as final prompt in your work-iteration-prompts.md

**What It Does:**

- Analyzes all implementation files in workspace
- Identifies changes (features, fixes, docs)
- Updates requirements.md and tech-spec.md
- Ensures documentation stays in sync

## The .rdd.analyze Prompt

**Purpose:** Iterative requirement clarification and execution planning

**Location:** `.github/prompts/rdd.analyse-and-plan.prompt.md`

**When to Use:**

- At the start of a complex iteration
- When requirements are unclear
- To generate a detailed execution plan

> ⚠️ **WARNING:** This prompt is iterative and could consume multiple premium requests. Try to clarify the user story manually or outside VS Code with GitHub Copilot first.

**What It Does:**

1. Reads your user story from `.rdd-docs/user-story.md`
2. Guides you through requirement clarification using a state-based workflow
3. Generates a Requirements Questionnaire with multiple-choice questions
4. Creates a detailed execution plan with sequenced prompts
5. Ensures all clarity criteria are met before implementation

**State-Based Workflow:**

- States 1-4: Collect main questions (What, Why, Acceptance Criteria, Other Considerations)
- State 5: Generate Requirements Questionnaire
- State 6-7: Answer questions and confirm completeness
- State 8-9: Generate and refine Execution Plan

> **Tip:** Always run .rdd.update before completing an iteration to keep your project documentation accurate and synchronized with implementation changes.

# Completing Your Iteration

When all prompts are executed and documentation is updated, complete the iteration using menu option 3: **Complete current iteration**. You must be on a feature branch (NOT on default branch) and the workspace must NOT be empty.

### Workspace Archiving

All files from `.rdd-docs/workspace/` are copied to `.rdd-docs/archive/<branch-name>/`. Archive metadata file created with timestamp, branch name, author, and commit info.

### Push to Remote

If not in local-only mode, you're prompted to push branch to origin. Displays reminder to create pull request on GitHub.

1

2

3

4

### Automatic Commit

All changes committed with message: "Completing work on <branch-name>". Includes code changes, documentation updates, and workspace files.

### Return to Default

Checks out the default branch and clears workspace, making it ready for the next iteration.

## What Happens Next

### 1. Create Pull Request

Use GitHub web interface to create a PR for your completed work. This happens outside the RDD framework.

### 2. Request Code Review

Assign team members to review your changes. Wait for approval before proceeding.

### 3. Merge & Cleanup

After PR approval and merge, use menu option 4 to delete the merged branch locally and keep your repository clean.