Network Working Group
Internet-Draft
Intended status: Informational
Expires: 1 December 2026

A. Dulaunoy
A. Iklody
CIRCL

30 May 2026

Programming Methodology Framework aka PMF
draft-dulaunoy-programming-methodology-framework-03

Abstract

This document describes the Programming Methodology Framework, also known as the PMF methodology. The methodology is based on the manifesto written by Zed A. Shaw [PROGRAMMING-MF-MANIFESTO], which describes a natural approach to software engineering with a strong focus on the act of programming. The PMF methodology uses a neutral name to provide a non-partisan reference for official engineering or project documents describing one of the most widely used software engineering methodologies.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 1 December 2026.

Copyright Notice

Table of Contents

1. Introduction

In 2011, Zed A. Shaw published a blog post which describes:

{align="left"} I think I'm going to create the ultimate software development methodology. It'll be revolutionary compared to others because it will focus on the one thing that gets software done. Its entire focus will be this one, glorious, completely useful activity for writing software. It's name even embodies the perfection of this programming methodology.

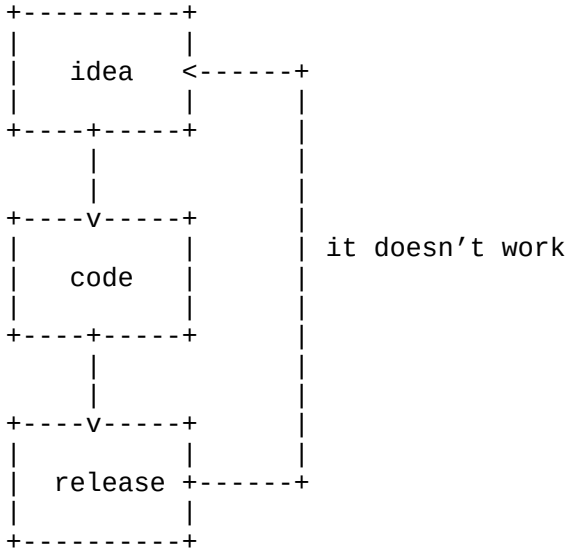
The PMF methodology was later published as a manifesto [PROGRAMMING-MF-MANIFESTO]. The manifesto clearly describes the focus on programming as a way to avoid surrounding management overhead and pivot towards software delivery. [THE-TAO-OF-PROGRAMMING] describes similar methodologies that strongly focus on coding, scripting, and programming.

The overall concept of PMF methodology follows the following process:

* Compose a list of features to be implemented and use an iterative programming process.
* Enhance said features using trial and error programming.
* Orchestrate your testing and integration processes using pragmatic programming.

Repeat the above process until the software is delivered.

A simplified overview of the process can be described as follows:



The PMF methodology covers both open source and proprietary software and can be used interchangeably in either context.

1.1. Management and PMF methodology

A simple management process is a requirement of the PMF methodology. The management process is meant to complement, rather than interfere with, the programming aspect, and it serves solely to support the PMF methodology.

* Collect requirements of the users/customers.
* Provide programmers with the desired requirements.
* Review whether the software to be delivered matches the requirements.

1.2. Swearwords and Software Engineering

PMF recognises that the vocabulary found in source code, comments, commit messages, and adjacent engineering conversations can carry useful information about how programmers experience the code they maintain. Strehmel’s empirical study of open-source C repositories compared projects containing English swearwords with projects that did not contain them and reported that the former set showed significantly better adherence to coding standards under several statistical tests [SWEARWORDS-CODE-QUALITY].

This observation is important for PMF because it treats apparently informal programmer expression as a signal that can be studied rather than dismissed. Swearwords can indicate emotional involvement, frustration with unnecessary complexity, or unusually direct feedback about code paths that deserve attention. In practical software engineering, such signals can help teams identify confusing interfaces, fragile components, and places where programmers have invested enough attention to leave candid warnings for future readers.

The correlation MUST NOT be interpreted as a recommendation to add swearwords to source code to improve quality. PMF instead recommends that teams preserve useful technical context, review emotionally charged comments with empathy, and turn the underlying engineering signal into tests, refactoring, documentation, or clearer interfaces.

1.3. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Security Considerations

Secure and defensive programming can only come from practicing programming, and this also includes the act of simplifying or removing code to reduce the attack surface.

3. Acknowledgements

The authors wish to thank all the programmers who program.

4. References

4.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

4.2. Informative References

[PROGRAMMING-MF-MANIFESTO] Shaw, Z. A., "Programming Motherfucker, do you speak it?", <http://programming-motherfucker.com>.

[SWEARWORDS-CODE-QUALITY]
Strehmel, J., "Is there a Correlation between the Use of Swearwords and Code Quality in Open Source Code?", 2023, <https://cme.h-its.org/exelixis/pubs/JanThesis.pdf>.

[THE-TAO-OF-PROGRAMMING]
James, G., "The Tao of Programming", <http://www.mit.edu/~xela/tao.html>.

Authors' Addresses

Alexandre Dulaunoy
Computer Incident Response Center Luxembourg
122, rue Adolphe Fischer
L-L-1521 Luxembourg
Luxembourg
Phone: +352 247 88444
Email: alexandre.dulaunoy@circl.lu

Andras Iklody
Computer Incident Response Center Luxembourg
122, rue Adolphe Fischer
L-L-1521 Luxembourg
Luxembourg
Phone: +352 247 88444
Email: andras.iklody@circl.lu