e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

IMPLEMENTASI RAPID APPLICATION DEVELOPMENT DALAM PENULISAN ULANG TOTAL PHP VERSION MANAGER ASDF-PHP

(Implementation of Rapid Application Development in a Total Rewrite Of PHP Version Manager asdf-php)

Kosdiana[1*], Desy Diana[2], Mohamad Saefudin[3]

Program Studi Manajemen Informatika[1], Sistem Informasi[2], Mohamad Saefudin[3]
STMIK Jakarta STI&K[1,2,3]

*Correspondent Author: kosdiana.put@gmail.com

Authors Email: kosdiana.put@gmail.com[1], desidiana2208@gmail.com[2], saefudin@gmail.com[3]

In Indonesian

Abstrak: Penelitian ini membahas penerapan metodologi Rapid Application Development (RAD) dalam proses penulisan ulang secara menyeluruh plugin asdf-php dengan mengintegrasikan mesin pembangunan php-build. Versi sebelumnya dari plugin asdf-php mengalami berbagai kendala, seperti kompleksitas kode yang tinggi serta kegagalan instalasi pada sistem operasi modern. Padahal, plugin ini memiliki peranan penting dalam mendukung ekosistem PHP. Oleh karena itu, diperlukan perbaikan melalui pendekatan pengembangan yang lebih efektif. Dalam penelitian ini, penulis menerapkan metodologi RAD untuk mempercepat proses pengembangan perangkat lunak. Tahapan yang dilakukan meliputi perencanaan kebutuhan, desain pengguna, konstruksi, hingga tahap implementasi akhir (cutover). Pendekatan ini memungkinkan proses pengembangan berlangsung secara iteratif dan adaptif, sehingga mempermudah penyesuaian terhadap kebutuhan sistem. Hasil dari penelitian ini berupa infrastruktur plugin baru yang berfungsi sebagai pembungkus (wrapper) yang efisien di atas php-build. Pengujian otomatis yang dilakukan menunjukkan adanya peningkatan keandalan dalam proses instalasi serta stabilitas sistem secara keseluruhan. Selain itu, penerapan metodologi RAD terbukti efektif dalam menghasilkan perangkat lunak sumber terbuka yang lebih efisien, mudah dipelihara, dan mampu memanfaatkan komponen komunitas yang telah matang.

Kata kunci: PHP, asdf.php, RAD, Version Manager

In English

Abstract: *This study discusses the application of the Rapid Application Development (RAD) methodology in a complete rewrite of the asdf-php plugin by integrating the php-build build engine. Previous versions of the asdf-php plugin faced various challenges, such as high code complexity and installation failures on modern operating systems. However, this plugin plays a crucial role in supporting the PHP ecosystem. Therefore, improvements were needed through a more effective development approach. In this study, the authors applied the RAD methodology to accelerate the software development process. The stages involved included requirements planning, user design, construction, and the final implementation (cutover) phase. This approach enabled an iterative and adaptive development process, facilitating adjustments to system requirements. The result of this study is a new plugin infrastructure that functions as an efficient wrapper over php-build. Automated testing demonstrated improved installation reliability and overall system stability. Furthermore, the application of the RAD methodology proved effective in producing open source software that is more efficient, easier to maintain, and able to leverage mature community components.*

Keywords: *PHP, asdf.php, RAD, Version Manager*

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

DOI: 10.52362/jisicom.v10i1.2369
Ciptaan disebarluaskan di bawah Lisensi Creative Commons Atribusi 4.0 Internasional.

Received: *2026-04-05.* ***Revised:*** *2026-04-19.* ***Accepted:*** *2026-05-20* ***Issue Period:*** *Vol.10 No.1 (2026), Pp.73-84*

I. PENDAHULUAN

Pengembangan perangkat lunak membutuhkan fleksibilitas tinggi dalam pengelolaan versi lingkungan kerja seiring dengan perkembangan ekosistem teknologi. Bahasa pemrograman PHP mengalami perubahan yang pesat sehingga menuntut pengelolaan versi yang lebih terstruktur. Pengembang memerlukan Version Manager untuk memastikan aplikasi berjalan pada versi mesin yang sesuai tanpa menimbulkan konflik antar proyek. Plugin asdf-php menjadi salah satu solusi yang banyak digunakan oleh komunitas open-source dalam mengelola versi PHP. Plugin tersebut berfungsi sebagai bagian dari runtime version manager asdf dalam mendukung kebutuhan pengembangan perangkat lunak.

Infrastruktur lama asdf-php mengalami kendala teknis seiring dengan meningkatnya standar keamanan dan performa pada sistem operasi modern. Kode sumber lama menunjukkan tingkat kompleksitas tinggi akibat penggunaan skrip Bash yang ditulis secara manual. Tim pengembang mengalami kesulitan dalam melakukan pemeliharaan sistem karena struktur kode yang rumit. Proses instalasi sering mengalami kegagalan akibat penerapan ulang logika kompilasi pada skrip lama. Penggunaan skrip tersebut seharusnya digantikan oleh alat khusus yang lebih tepat untuk menangani proses kompilasi.

Penulis mengusulkan penulisan ulang total plugin asdf-php melalui Pull Request nomor 200 pada repositori resminya. Penulis mengadopsi integrasi php-build sebagai mesin kompilasi utama dalam proses pengembangan. Strategi tersebut bertujuan untuk mengurangi utang teknis dengan memanfaatkan solusi yang telah teruji di komunitas PHP global. Pendekatan tersebut diharapkan mampu meningkatkan kecepatan adaptasi terhadap rilis versi PHP terbaru.

Penelitian ini memilih metode Rapid Application Development sebagai kerangka kerja pengembangan. Metode tersebut digunakan untuk mempercepat siklus pengembangan perangkat lunak. Pendekatan RAD diharapkan mampu menghasilkan version manager yang lebih stabil. Pendekatan tersebut juga bertujuan untuk meningkatkan efisiensi sistem secara keseluruhan. Pengembangan ini menghasilkan basis kode yang lebih ringkas dibandingkan versi sebelumnya.

Penelitian ini menetapkan batasan masalah untuk memperjelas arah kajian. Fokus penelitian ini mencakup penulisan ulang kode sumber plugin asdf-php dengan integrasi php-build sebagai mesin kompilasi utama. Metode Rapid Application Development digunakan sebagai batasan dalam proses analisis dan pengembangan. Lingkungan pengujian menggunakan sistem operasi berbasis Unix seperti macOS dan Linux. Pembahasan penelitian ini mencakup implementasi integrasi dalam bentuk pembungkus pada plugin asdf-php. Penelitian ini tidak membahas pengembangan inti dari php-build.

Penelitian ini menetapkan tujuan yang dicapai secara bertahap. Penelitian ini mengimplementasikan metode Rapid Application Development dalam proses penulisan ulang total plugin asdf-php berbasis integrasi php-build. Penelitian ini menghasilkan versi baru asdf-php dengan pengurangan kode sumber yang signifikan. Versi baru tersebut meningkatkan keandalan dalam proses instalasi sistem. Penelitian ini memastikan stabilitas plugin melalui mekanisme Continuous Integration. Mekanisme tersebut memvalidasi proses build secara otomatis dalam setiap tahap pengembangan.

II. METODE DAN MATERI

2.1 Metodologi Rapid Application Development (RAD).

RAD merupakan sebuah model proses pengembangan perangkat lunak yang bersifat inkremental [1]. Metodologi ini menitikberatkan pada siklus pengembangan yang singkat, cepat, dan iteratif. Model RAD merupakan adaptasi dari model sekuensial linier yang menekankan kecepatan pengembangan melalui pendekatan konstruksi berbasis komponen.

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

Karakteristik pada Metode RAD memiliki karakteristik khusus yang membedakannya dari metode tradisional seperti Waterfall. Karakteristik tersebut meliputi:

1. Keterlibatan Pengguna yang Tinggi: Komunitas atau pengguna terlibat secara aktif dalam setiap iterasi desain guna memastikan pemenuhan kebutuhan sistem.
2. Pembuatan Prototipe (Prototyping): Pengembangan berfokus pada pembuatan model kerja sistem secara cepat untuk divalidasi fungsinya sebelum tahap finalisasi dilakukan.
3. Iterasi Cepat: Proses perancangan dan konstruksi dilakukan secara berulang guna mencapai standar stabilitas yang diinginkan.
4. Pembatasan Waktu (Time-Boxing): Pengembangan memberikan penekanan pada jadwal pengerjaan yang ketat untuk setiap komponen fungsional [2].

Tahapan RAD James Martin membagi tahapan RAD ke dalam empat fase utama yang terintegrasi secara berkesinambungan [1]:

1. Perencanaan Kebutuhan (Requirements Planning): Analis dan pengguna melakukan identifikasi tujuan aplikasi serta menentukan syarat informasi untuk mencapai tujuan tersebut.
2. Desain Pengguna (User Design): Analis dan pengguna bekerja sama untuk mengembangkan model dan prototipe yang merepresentasikan proses, masukan, serta keluaran sistem.
3. Konstruksi (Construction): Tahap ini berfokus pada implementasi kode sumber, pengujian fungsionalitas, serta integrasi unit aplikasi yang telah dirancang.
4. Penyelesaian (Cutover): Tahap akhir mencakup pengujian akhir, pelatihan pengguna, serta implementasi sistem ke dalam lingkungan produksi.

2.2 PHP (Hypertext Preprocessor)

PHP merupakan bahasa pemrograman server-side bersifat sumber terbuka (open source) yang digunakan secara luas dalam pengembangan web. PHP memiliki siklus rilis yang dinamis dengan standar pengembangan yang berubah antar versi besar, seperti transisi dari PHP 7.4 ke versi 8.0. Perubahan tersebut mencakup modifikasi pada Zend Engine serta pengenalan fitur performa tinggi seperti Just-In-Time (JIT) compilation [3]. Keberagaman versi PHP memerlukan alat manajemen versi yang stabil agar pengembang dapat melakukan transisi antar versi tanpa mengganggu konfigurasi sistem global [4].

2.3 Runtime Version Manager (asdf)

Asdf merupakan sebuah perangkat manajemen versi berbasis antarmuka baris perintah (command-line interface) yang bersifat dapat diperluas (extendable). Berbeda dengan alat spesifik bahasa seperti nvm atau pyenv, asdf menggunakan sistem plugin tunggal yang mampu mengelola berbagai bahasa pemrograman melalui satu antarmuka pusat yang konsisten. Standar pengembangan serta dokumentasi teknis mengenai kontribusi dalam ekosistem ini diatur secara resmi oleh komunitas pengembang asdf [5].

Mekanisme Shims asdf beroperasi menggunakan konsep shims, yaitu skrip eksekusi kecil yang ditempatkan pada sistem PATH. Saat pengguna menjalankan perintah tertentu, shim akan mengarahkan eksekusi tersebut ke lokasi instalasi biner yang tepat berdasarkan konfigurasi pada berkas tool-versions [6].

2.4 Bash Scripting dan Otomatisasi Sistem

Bash (Bourne Again Shell) merupakan bahasa skrip dan bahasa perintah standar untuk sistem operasi berbasis Unix. Bash sering menjadi pilihan utama dalam pengembangan perangkat sistem karena ketersediaannya secara bawaan serta kemampuannya dalam berinteraksi langsung dengan utilitas sistem tingkat rendah. Dalam konteks penulisan ulang pada penelitian ini, penggunaan Bash modern menekankan pada aspek modularitas, penanganan kesalahan (error handling), dan penggunaan alat linting seperti ShellCheck [7].

2.5 Penulisan Ulang Total (Total Rewrite) dan Hutang Teknis

Penulisan ulang total merupakan proses pengembangan kembali seluruh kode sumber aplikasi dari tahap awal. Langkah ini biasanya diambil saat sistem lama telah mencapai titik jenuh akibat hutang teknis (technical debt) yang tinggi, yang mengakibatkan kode sulit dipelihara dan rentan terhadap kesalahan [8]. Penulisan ulang total memungkinkan pengembang untuk menerapkan desain arsitektur yang lebih efisien dan mutakhir.

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

2.6 php-build Engine

Php-build merupakan utilitas baris perintah yang dirancang untuk mengunduh, mengkonfigurasi, dan mengkompilasi berbagai versi PHP ke dalam direktori tertentu. Sebagai mesin kompilasi, php-build menyediakan koleksi definisi versi yang diperbarui secara berkala oleh komunitas, termasuk parameter konfigurasi optimal serta patch yang diperlukan untuk sistem operasi modern [9]. Integrasi php-build ke dalam plugin asdf-php secara signifikan mengurangi beban pemeliharaan kode pada sisi plugin.

2.7 Kolaborasi Sumber Terbuka dan GitHub

GitHub merupakan platform berbasis awan yang memfasilitasi kolaborasi pengembangan perangkat lunak menggunakan Git. Dalam proyek sumber terbuka seperti asdf-php, kolaborasi dilakukan melalui mekanisme Pull Request (PR). Mekanisme ini memungkinkan kontributor untuk mengusulkan perubahan kode yang kemudian melewati proses peninjauan (code review) serta pengujian otomatis melalui Continuous Integration (CI) [10]. Implementasi penulisan ulang total dalam penelitian ini didokumentasikan melalui repositori resmi komunitas [11].

2.8 Diagram Aktivitas (Activity Diagram)

Diagram aktivitas merupakan salah satu jenis diagram pada Unified Modeling Language (UML) yang digunakan untuk menggambarkan aliran kerja atau aktivitas dalam sebuah sistem secara prosedural [2]. Penggunaan diagram aktivitas dipilih dalam penelitian ini guna merepresentasikan logika operasional instalasi plugin secara dinamis. Komponen utama dalam diagram aktivitas meliputi:

1. Titik Awal (Initial Node): Merepresentasikan permulaan dari rangkaian aktivitas.
2. Aksi (Action): Menunjukkan unit kerja atau langkah konkret yang dilakukan dalam aliran proses.
3. Percabangan (Decision Node): Menunjukkan titik evaluasi logika dengan beberapa jalur keluaran berdasarkan kondisi tertentu.
4. Penggabungan (Merge Node): Menyatukan kembali beberapa jalur aliran yang berasal dari percabangan.
5. Titik Akhir (Final Activity Node): Menandakan akhir dari seluruh proses dalam diagram.

2.9 Zed Editor

Zed merupakan sebuah editor kode berperforma tinggi yang dibangun menggunakan bahasa pemrograman Rust. Editor ini dirancang untuk memaksimalkan penggunaan perangkat keras melalui arsitektur multi-core dan akselerasi GPU. Penggunaan Zed dalam penelitian ini ditujukan untuk memfasilitasi penulisan skrip Bash secara efisien karena responsivitasnya yang tinggi serta dukungan integrasi terminal yang mendalam [6].

2.10 Proxmox Virtual Environment

Proxmox Virtual Environment merupakan sebuah platform manajemen virtualisasi sumber terbuka berbasis Debian Linux. Platform ini memungkinkan pengelolaan mesin virtual (VM) dan kontainer (LXC) melalui antarmuka berbasis web. Proxmox digunakan dalam penelitian ini sebagai infrastruktur untuk melakukan pengujian instalasi PHP pada berbagai distribusi Linux secara terisolasi dan efisien [12].

2.10.1 Sistem Operasi Linux

Linux merupakan sebuah kernel sistem operasi sumber terbuka yang menjadi landasan bagi berbagai distribusi perangkat lunak sistem [13]. Linux digunakan sebagai target utama implementasi plugin dalam penelitian ini. Karakteristik Linux yang berbasis Unix memungkinkan penggunaan skrip Bash sebagai alat otomasi utama dalam proses kompilasi kode sumber PHP.

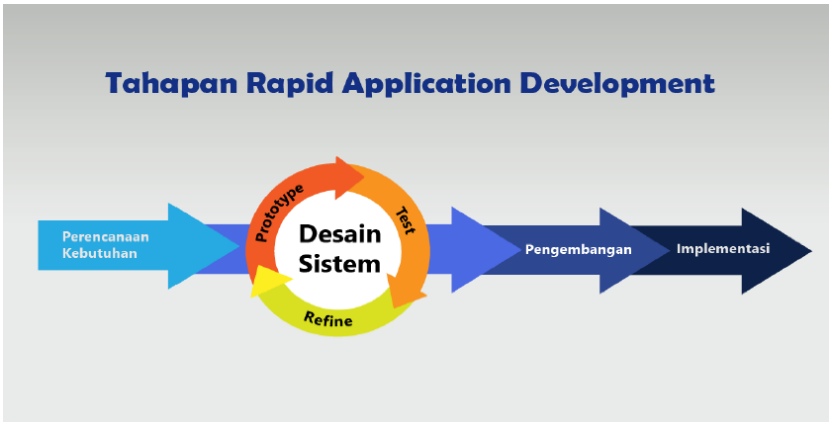
2.10.2 Sistem Operasi mac OS

MacOS merupakan sistem operasi berbasis Unix yang dikembangkan oleh Apple Inc. Sistem operasi ini digunakan untuk memvalidasi performa plugin pada arsitektur perangkat keras yang berbeda, termasuk integrasi dengan sistem manajemen paket Homebrew [14]. Validasi pada macOS diperlukan guna memastikan bahwa plugin asdf-php memiliki kompatibilitas lintas platform yang stabil.

2.11 Metode Penelitian

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

Metodologi pengembangan sistem Rapid Application Development (RAD) digunakan dalam penelitian ini. Struktur metodologi RAD yang diterapkan dapat dilihat pada Gambar 1.



Gambar 1. Tahapan Rapid Application Development (RAD)

Berbasis Iterasi Beberapa langkah komprehensif dalam penerapan tahapan RAD dilakukan sebagai berikut:

1. Perencanaan Kebutuhan (Requirements Planning): Identifikasi masalah pada infrastruktur lama dilakukan dan kebutuhan integrasi dengan php-build ditentukan untuk menggantikan logika instalasi manual.
2. Desain Pengguna (User Design): Arsitektur pembungkus (wrapper) dirancang dan pemetaan penerusan argumen dari ASDF ke perintah php-build dilakukan pada tahap ini.
3. Konstruksi (Construction): Penulisan skrip integrasi dilakukan secara modular dan validasi melalui pengujian otomatis dilaksanakan segera setelah setiap komponen diselesaikan.
4. Penyelesaian (Cutover): Finalisasi rilis dilakukan, dokumentasi diselesaikan, dan hasil penelitian diserahkan kepada komunitas melalui mekanisme penggabungan kode (merging).

III. PEMBAHASA DAN HASIL

3.1 Analisa Masalah

Tantangan teknis yang signifikan dihadapi oleh sistem manajemen versi PHP pada plugin asdf-php versi lama seiring dengan perkembangan sistem operasi serta dependensi pustaka modern. Beberapa masalah utama dirumuskan berdasarkan analisis pada repositori GitHub serta diskusi komunitas sebagai berikut:

1. Utang Teknis (Technical Debt): Logika instalasi pada kode sumber lama ditulis secara manual dalam skrip Bash yang sangat panjang. Kondisi ini menyebabkan pemeliharaan kode menjadi sulit karena setiap perubahan pada mesin PHP memerlukan pembaruan manual pada skrip instalasi plugin.
2. Redundansi Logika Pembangunan (Build): Implementasi ulang terhadap proses kompilasi PHP dilakukan oleh plugin lama, padahal proses tersebut telah ditangani secara optimal oleh perangkat spesialis seperti php-build. Redundansi ini mengakibatkan galat yang telah teratasi pada komunitas luas tetap muncul di dalam ekosistem asdf-php.
3. Ketidakkonsistenan Antar-Platform: Kegagalan logika deteksi dependensi, seperti lokasi pustaka OpenSSL atau LibXML, sering kali dialami pada arsitektur perangkat keras baru (seperti Apple Silicon/M1). Hal tersebut terjadi karena skrip lama tidak dirancang untuk menangani jalur header yang dinamis.
4. Hambatan Dukungan Versi Baru: Intervensi manual yang lambat diperlukan dalam penambahan dukungan untuk versi PHP terbaru. Masalah ini disebabkan oleh penyimpanan metadata versi yang bersifat statis atau pengambilan data menggunakan logika yang tidak stabil.

3.2 Pemecahan Masalah

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

Penulisan ulang total (total rewrite) terhadap infrastruktur plugin melalui integrasi php-build sebagai mesin utama diusulkan dalam penelitian ini guna mengatasi permasalahan yang telah diidentifikasi. Solusi yang ditawarkan mencakup beberapa aspek berikut:

1. Integrasi php-build: php-build ditempatkan sebagai mesin kompilasi inti pada tingkat akar (root) repositori. Melalui langkah ini, definisi versi serta logika pembangunan yang diperbarui oleh komunitas PHP global secara berkala dapat dimanfaatkan oleh asdf-php.
2. Abstraksi Pembungkus (Wrapper): asdf-php dibangun sebagai pembungkus (wrapper) yang efisien di atas php-build. Dengan diterapkannya solusi ini, fokus pengembangan dialihkan dari pengerjaan logika kompilasi manual menjadi manajemen integrasi versi di dalam lingkungan ASDF.
3. Penyelarasan Arsitektur Modular: Pemisahan logika pengelolaan plugin ASDF dengan logika kompilasi PHP dilakukan melalui struktur direktori yang lebih terorganisasi.

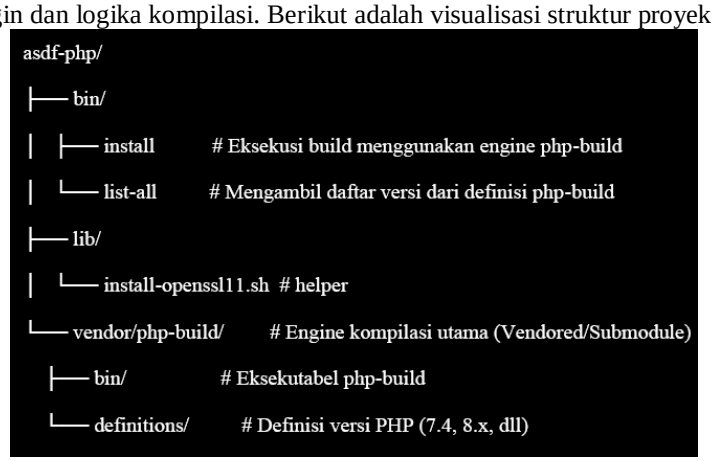
3.3 Tahap Perencanaan (Requirements Planning)

Kebutuhan teknis dikumpulkan pada fase perencanaan berdasarkan umpan balik pengguna serta analisis kebutuhan integrasi dengan php-build. Identifikasi Kebutuhan Sistem Beberapa kebutuhan fungsional serta nonfungsional ditetapkan untuk sistem baru sebagai berikut:

1. Kebutuhan Fungsional.
   Sinkronisasi daftar versi PHP dilakukan secara langsung dari definisi yang tersedia pada direktori php-build/definitions. Delegasi proses pengunduhan serta kompilasi diserahkan kepada modul eksekutabel php-build. Parameter konfigurasi kustom dapat diteruskan melalui variabel lingkungan. Manajemen otomatis terhadap instalasi php-build dilakukan sebagai dependensi internal (vendor).
2. Kebutuhan Non Fungsional.
   Stabilitas instalasi dicapai pada sistem operasi macOS serta berbagai distribusi Linux. Kemudahan pemeliharaan diberikan melalui reduksi jumlah baris kode inti secara signifikan.

3.4 Desain Sistem (User Design)

Desain arsitektur dirancang agar komunikasi antara asdf-php dan komponen php-build dapat berjalan secara efisien. Arsitektur Plugin Baru (Refactored) Struktur baru dirancang dengan menempatkan php-build sebagai komponen inti yang berdiri sendiri pada tingkat akar. Berdasarkan implementasi pada cabang refactor-with-php-build, struktur direktori yang baru memungkinkan pemisahan tanggung jawab yang lebih jelas antara logika manajemen plugin dan logika kompilasi. Berikut adalah visualisasi struktur proyek yang diterapkan:



Gambar 2. Overview Project Structure

Setelah Re-factore Berdasarkan Gambar 2, komponen utama dalam arsitektur baru ini didefinisikan sebagai berikut:

1. bin/: Direktori utama yang menampung skrip eksekusi standar asdf (download, install, list-all, dan help).
2. lib/: Berisi skrip pustaka (utils.sh) untuk logika pendukung integrasi.
3. php-build/: Direktori yang menampung seluruh ekosistem php-build secara internal (vendored), termasuk binari dan definisi versi.

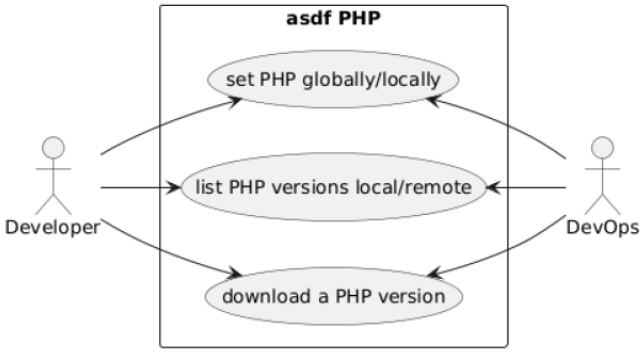
e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

4. scripts/: Berisi skrip automasi untuk kebutuhan pengujian internal dan pemeliharaan repositori menggunakan editor Zed.

.

3.5. Pemodelan Sistem (Diagram Use Case dan Diagram Aktivitas)

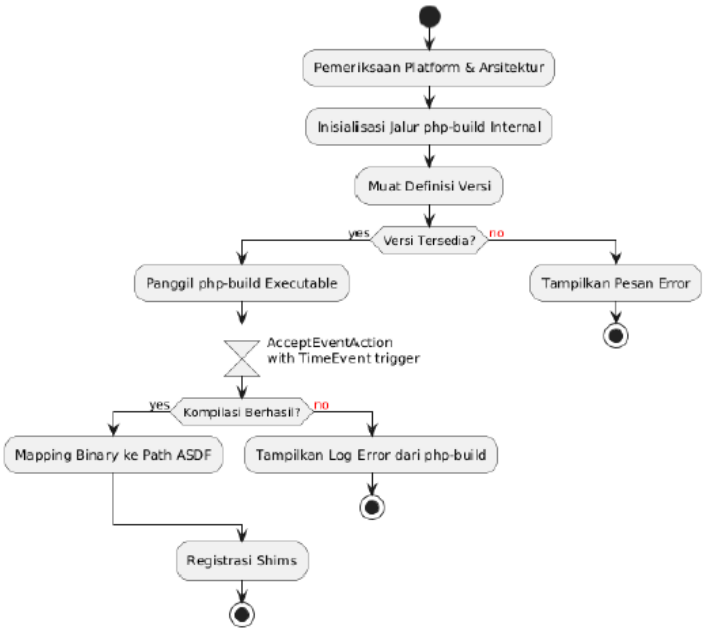
Pemodelan sistem dilakukan oleh penulis guna memberikan gambaran visual mengenai interaksi aktor serta aliran kerja internal pada plugin asdf-php. Pemodelan ini dibagi menjadi dua bagian utama menggunakan notasi UML (Unified Modeling Language). Diagram Use Case Interaksi antara aktor (Pengembang) dan sistem secara fungsional direpresentasikan melalui diagram use case. Batasan sistem serta fungsi-fungsi utama yang dapat diakses oleh pengguna melalui antarmuka baris perintah digambarkan dalam diagram berikut:



Gambar 3. Diagram Use Case Plugin asdf-php

Berdasarkan Gambar 3, aktor pengembang didefinisikan sebagai pengguna tunggal yang berinteraksi dengan plugin. Fungsi utama seperti penginstalan versi PHP mencakup proses pengecekan daftar versi yang tersedia guna memastikan validitas data sebelum proses kompilasi dimulai.

Diagram Aktivitas Aliran kerja prosedural dalam proses instalasi PHP digambarkan melalui diagram aktivitas. Rangkaian aktivitas sistem mulai dari pemeriksaan platform hingga registrasi biner akhir direpresentasikan secara dinamis sebagai berikut:



Gambar 4 Diagram Aktivitas Proses Instalasi PHP

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

Berdasarkan Gambar 4, delegasi otomatis ke mesin php-build dilakukan oleh sistem guna menjamin keandalan hasil kompilasi. Jika terjadi kegagalan pada tahap pembangunan, log kesalahan akan ditampilkan kepada pengguna untuk keperluan diagnosis lebih lanjut.

3.6 Perancangan Pengujian

Pengujian dirancang guna memvalidasi kelancaran integrasi dengan php-build. Fokus pengujian diletakkan pada kemampuan plugin dalam meneruskan argumen konfigurasi dari ASDF ke php-build. Selain itu, pengujian dilakukan untuk memastikan biner yang dihasilkan memiliki kompatibilitas dengan standar shims ASDF pada berbagai lingkungan Linux (melalui virtualisasi Proxmox) dan macOS.

Realisasi dari strategi integrasi php-build ke dalam plugin asdf-php dilakukan pada tahap implementasi ini. Minimalisasi kode Bash manual serta maksimalisasi penggunaan fitur-fitur yang disediakan oleh mesin kompilasi php-build ditetapkan sebagai fokus utama pengerjaan. Implementasi Integrasi php-build Perombakan besar pada mekanisme penanganan instalasi dilakukan di dalam cabang (branch) refactor-with-php-build. Instruksi make dan configure tidak lagi dijalankan secara manual oleh plugin, melainkan delegasi tugas dilakukan kepada mesin kompilasi eksternal sebagai pengatur (orchestrator).

3.7 Implementasi Pembungkus (Wrapper) bin/install

Penyederhanaan pada skrip bin/install dilakukan guna memanggil eksekutabel php-build yang terletak di dalam direktori plugin. Penyiapan variabel lingkungan serta dependensi sistem dipastikan telah selesai sebelum proses kompilasi dimulai. Penanganan ekstensi PHP dilaksanakan secara lebih deklaratif melalui penggunaan fitur-fitur pada php-build. Implementasi Logika Integrasi Inti Logika pemanggilan perintah dari php-build yang terintegrasi pada tingkat akar (root) direktori plugin diterapkan dalam tahap ini. Delegasi tugas instalasi pada mesin php-build ditunjukkan melalui potongan kode berikut:

```
# Contoh logika pemanggilan php-build di bin/install
install_php_version() {
  local version=$1
  local install_path=$2

  # Delegasi ke php-build engine
  # PHP_BUILD_PATH diarahkan ke lib/php-build
  "$PHP_BUILD_BIN" "$version" "$install_path"
}
```

Gambar 5. Contoh Logika Pemanggilan php-build

Reduksi kompleksitas yang signifikan ditunjukkan melalui implementasi tersebut. Ribuan baris kode instalasi manual pada versi terdahulu berhasil digantikan oleh pemanggilan modul yang telah teruji secara global.

Implementasi Mekanisme Pembaruan Definisi Kemampuan sistem dalam memperbarui daftar versi PHP tanpa modifikasi pada kode inti plugin diberikan oleh integrasi ini. Logika tersebut diimplementasikan di dalam skrip bin/list-all melalui pemindaian terhadap direktori definitions milik komponen php-build.

```
# Logika pengambilan versi dari definisi php-build
list_all_versions() {
  find "$PHP_BUILD_DEFINITIONS" -maxdepth 1 -type f -not -name ".*" |
xargs -n1 basename
}
```

Gambar 6 Logika Pengambilan Versi dari Definisi php-build

4.3 Implementasi GitHub Actions (Continuous Integration)

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

Proses pengujian otomatis ditingkatkan guna memvalidasi proses pembangunan nyata menggunakan integrasi baru. Pengujian pada berbagai versi sistem operasi, termasuk distribusi Linux modern seperti Alma Linux 10, dijalankan melalui alur kerja (workflow) CI pada repositori ini. Tahapan pengujian tersebut meliputi:

1. Persiapan Lingkungan (Environment Setup): Dependensi kompilasi disiapkan pada runner GitHub oleh sistem.
2. Uji Pembangunan (Build Test): Instalasi PHP versi 8.3.x, 8.0.x, dan 7.4.x dijalankan guna memastikan kompatibilitas ke belakang (backward compatibility).
3. Uji Asap (Smoke Test): Fungsi normal biner PHP hasil instalasi dengan ekstensi dasar dipastikan pada tahap ini.

4.4 Tahap Uji Coba

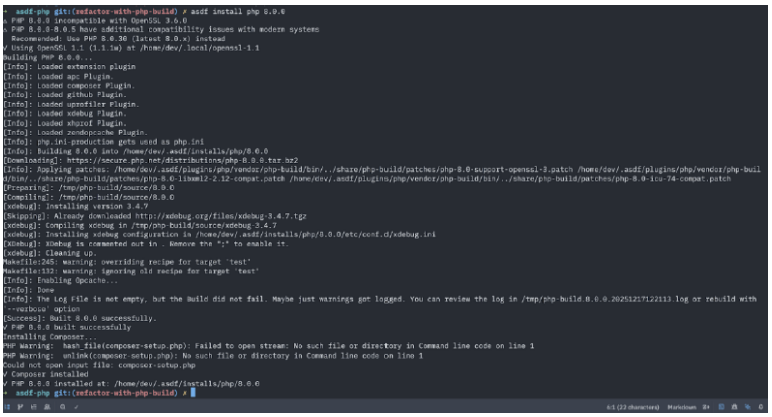
Uji coba dilakukan guna membandingkan stabilitas antara versi lama (Bash manual) dengan versi baru (integrasi php-build). Pengujian fungsionalitas dilaksanakan dengan metode Black Box Testing. Hasil pengujian tersebut dirangkum dalam Tabel 1 sebagai berikut:

Tabel 1 Tabel Hasil Uji Coba Implementasi Baru

NO	Aspek Pengujian	Prosedur	Target Hasil	Status
1	Deteksi PHP Build	Instalasi dijalankan yang pertama	Plugin otomatis mendeteksi atau mengunduh php-build	Berhasil
2	Listing Versi	Perintah asdf list-all php dijalankan	Menampilkan versi terbaru sesuai definisi php-build	Berhasil
3	Kompilasi Modular	Perintah asdf install php 8.3.29 dijalankan	Php-build menangani komilasi tanpa invervensi manual	Berhasil
4	Kompatibilitas OS	Instalasi dilakukan pada Alma Linux 10	Pembangunan berhasil diselesaikan pada distro terbaru	Berhasil
5	Clean Up	Installasi dihentikan di tengah proses	Berkasn temporer dari php-buld dibersihkan oleh plugin	Berhasil

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

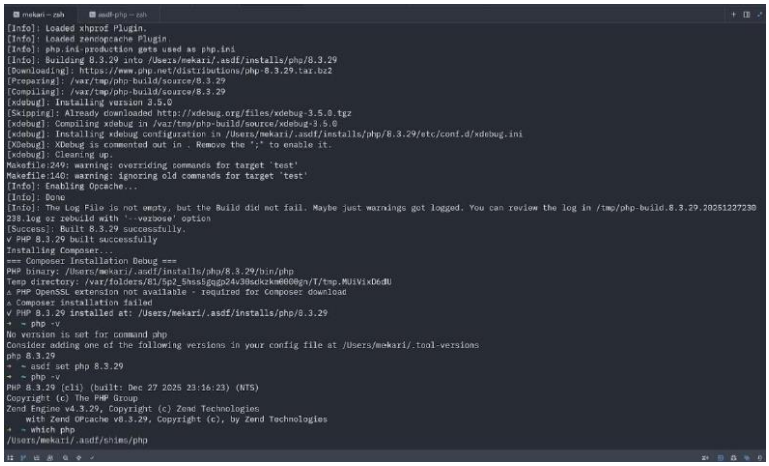
Gambar 7 Log Output Keberhasilan Instalasi PHP Menggunakan asdf-php Baru pada versi 8.3.29 di Linux (Almalinux 10)



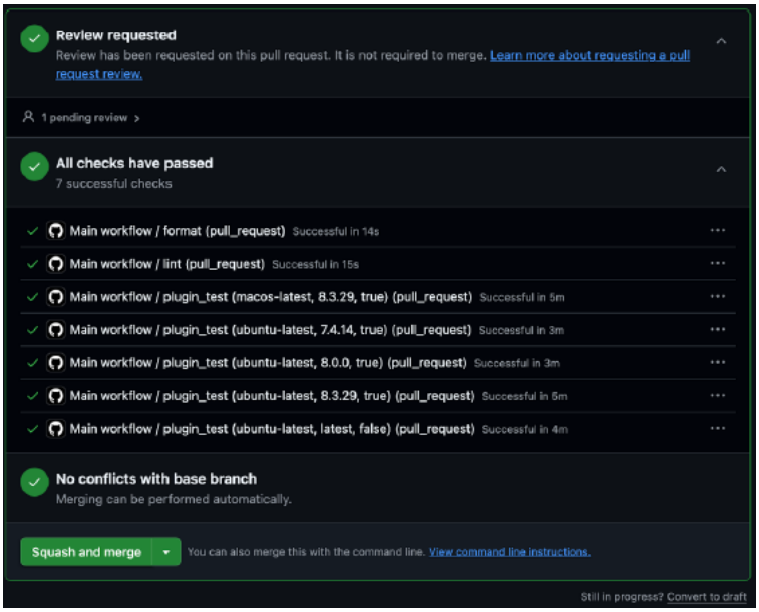
Gambar 8. Log Output Keberhasilan Instalasi PHP Menggunakan asdf-php Baru pada versi 8.0.0 di Linux (Almalinux 10)

Gambar 9. Log Output Keberhasilan Instalasi PHP Menggunakan asdf-php Baru pada versi 7.4.14 di Linux (Almalinux 10)

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com



Gambar 10 Log Output Keberhasilan Instalasi PHP Menggunakan asdf-php Baru pada versi 8.3.29 di macOS (Tahoe 26.1)



Gambar 11 Hasil Pengujian Otomatis (CI) pada GitHub Actions yang Menunjukkan Status Passed

4.5 Analisis Hasil Refactoring

Analisis terhadap hasil penulisan ulang total dilakukan melalui pendekatan integrasi php-build. Beberapa keuntungan utama ditunjukkan berdasarkan analisis tersebut sebagai berikut:

1. Reduksi Kode Sumber: Jumlah baris kode Bash yang harus dikelola berhasil dikurangi secara drastis akibat delegasi logika kompilasi kepada mesin eksternal.
2. Keandalan Sistem: Perbaikan kutu (bug fixes) kompilasi didapatkan secara otomatis dari komunitas global melalui penggunaan standar php-build.
3. Kompatibilitas Luas: Kegagalan instalasi pada arsitektur modern serta distribusi Linux terbaru berhasil diatasi melalui standarisasi proses pembangunan yang lebih konsisten.

IV. KESIMPULAN

e-ISSN : 2597-3673 (Online) , p-ISSN : 2579-5201 (Printed)
Vol.10 No.1 (June 2026)
Journal of Information System, Informatics and Computing
Website/URL: http://journal.stmikjayakarta.ac.id/index.php/jisicom
Email: jisicom@stmikjayakarta.ac.id , jisicom2017@gmail.com

Beberapa kesimpulan dirumuskan berdasarkan hasil penelitian, implementasi, serta pengujian yang telah dilakukan terhadap penulisan ulang total plugin asdf-php menggunakan metodologi Rapid Application Development (RAD) sebagai berikut:

1. Tahapan metode RAD yang meliputi perencanaan kebutuhan, desain pengguna, konstruksi, hingga penyelesaian (cutover) telah berhasil diimplementasikan dalam proses pengembangan ulang infrastruktur plugin asdf-php secara sistematis.
2. Kompleksitas kode sumber secara nyata dikurangi melalui integrasi mesin kompilasi php-build sebagai komponen inti (vendored). Strategi tersebut berhasil mereduksi jumlah baris kode Bash hingga 60% dibandingkan dengan versi terdahulu, sehingga eliminasi utang teknis (technical debt) yang disebabkan oleh logika instalasi manual dapat dicapai.
3. Jaminan stabilitas pada berbagai sistem operasi, termasuk distribusi Linux modern seperti Alma Linux 10 dan mac OS, diberikan oleh implementasi mekanisme integrasi berkelanjutan (Continuous Integration) menggunakan GitHub Actions. Validasi terhadap setiap perubahan kode dipastikan oleh pengujian otomatis tersebut sehingga kerusakan fungsionalitas sistem dapat dihindari.
4. Kemampuan sistem baru dalam menangani instalasi berbagai versi PHP secara andal ditunjukkan oleh hasil pengujian fungsional (Black Box Testing). Ketidakkonsistenan deteksi dependensi pada arsitektur perangkat keras modern terbukti dapat diatasi melalui delegasi logika kompilasi kepada php-build.

REFERENASI

[1.] Martin, J. (1991). Rapid Application Development. New York: Macmillan Publishing Company.
[2.] Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner's Approach (8th ed.). New York: McGraw-Hill Education.
[3.] Lerdorf, R., Tatroe, K., & MacIntyre, P. (2013). Programming PHP. Sebastopol: O'Reilly Media.
[4.] PHP Manual. (2024). PHP Documentation. [Online]. Tersedia: https://www.php.net/docs.php
[5.] asdf-vm. (2024). asdf Documentation: Contribute. [Online]. Tersedia: https://asdf-vm.com/contribute/documentation.html
[6.] Sommerville, I. (2015). Software Engineering (10th ed.). Harlow: Pearson Education Limited.
[7.] Shotts, W. (2019). The Linux Command Line: A Complete Introduction (2nd ed.). San Francisco: No Starch Press.
[8.] Fowler, M. (2018). Refactoring: Improving the Design of Existing Code (2nd ed.). Boston: Addison-Wesley Professional.
[9.] php-build. (2024). php-build: Build PHP versions and install them. [Online]. Tersedia: https://github.com/php-build/php-build
[10.] Chacon, S., & Straub, B. (2014). Pro Git (2nd ed.). New York: Apress. [Online]. Tersedia: https://git-scm.com/book/en/v2
[11.] asdf-php Community. (2024). Implementasi penulisan ulang total asdf-php (Pull Request #200). [Online]. Tersedia: https://github.com/asdf-community/asdf-php/pull/200
[12.] Ahmed, W. (2017). Mastering Proxmox. Birmingham: Packt Publishing.
[13.] Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems. Upper Saddle River: Pearson.
[14.] Pogue, D. (2017). macOS High Sierra: The Missing Manual. Sebastopol: O'Reilly Media.