Semester Long Internship Report

On

Tool Manager For eSim

Submitted by

Pyae Sone Khant Aung
Final Year Student
Bachelor of Engineering (Honours) in
Electronics and Communications Engineering
Myanmar Institute of Information Technology
Mandalay,Myanmar

Under the guidance of

Prof. Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

March, 2025

Acknowledgements

I would like to express my sincere gratitude to the FOSSEE team for providing me with the opportunity to contribute to the development of the open-source software eSim. Their support and guidance have been invaluable in enabling me to work on this project and enhance my technical skills.

First and foremost, I extend my deepest appreciation to **Prof. Kannan M. Moudgalya** for his invaluable guidance and unwavering support throughout the FOSSEE Winter Internship program. His mentorship has been instrumental in the successful completion of my work.

I am also grateful to the entire eSim team for their continuous assistance and for providing the necessary resources to complete this project. In particular, I would like to express my sincere thanks to **Mrs. Usha Viswanathan** and **Mrs. Vineeta Parmar** for their constant support and encouragement.

Furthermore, I would like to acknowledge my mentor, **Mr. Sumanto Kar**, for his dedicated supervision, continuous support, and willingness to share his expertise. His guidance has significantly enriched my learning experience and contributed to my professional growth.

I would also like to express my appreciation to my fellow interns for making this journey both educational and enjoyable. It has been a privilege to collaborate with them under the guidance of such experienced mentors.

Additionally, I extend my gratitude to the **Spoken Tutorial Lab** for providing a conducive environment to carry out my work. The facilities and resources available in the lab greatly facilitated my progress and contributed to the successful execution of my tasks.

Finally, I would like to thank everyone who contributed, either directly or indirectly, to the successful completion of my internship. I am truly grateful for their support and encouragement, and I hope to apply the knowledge and skills gained from this experience to make meaningful contributions in the future.

Contents

Chapter 1

Introduction

The Automated Tool Manager simplifies managing external tools and dependencies for eSim, an open-source EDA tool for circuit design and simulation. Manually handling tools like Ngspice and KiCad can be complex, involving installation, configuration, updates, and dependencies.

This project automates these tasks, reducing manual effort and ensuring system compatibility. By streamlining tool management, it enhances eSim's efficiency and usability, allowing engineers to focus on circuit design rather than software setup. Designed for reliability and ease of use, it benefits users of all experience levels.

1.1 Project Objectives

The objectives of the Tool Manager for eSim project are as follows:

- Tool Installation Management: Automates the download and installation of required tools, ensuring compatibility and correct versions.
- Update and Upgrade System: Provides functionality to check for and apply updates with minimal user effort.
- Configuration Handling: Automates tool configuration and manages paths and environment variables for seamless integration.
- Dependency Checker: Verifies and resolves missing dependencies while providing user feedback.
- User Interface: Offers a user-friendly interface to manage tools, view versions, and track updates.
- Additional Features: Supports multiple platforms and integrates with popular package managers.

1.2 Project Motivation

Managing external tools and dependencies is a crucial but challenging aspect of using eSim, requiring manual tasks like downloading, configuring, and updating tools while

ensuring system compatibility. These complexities can be overwhelming, especially for users with limited technical expertise, affecting productivity and focus on circuit design.

The Automated Tool Manager aims to eliminate these challenges by automating installation, configuration, updates, and dependency management. This reduces manual intervention, minimizes errors, and ensures a seamless experience for eSim users.

By simplifying tool management and enhancing usability, this project empowers engineers and researchers to focus on innovation and design, aligning with eSim's mission of fostering open-source solutions for the engineering community.

1.3 Project Outline

The Automated Tool Manager is introduced in the first chapter, highlighting its role in simplifying eSim tool management. It addresses challenges in manual installation, configuration, and updates, while also outlining the motivation and objectives behind automation.

Following this, the problem statement is discussed, detailing the complexities of manual tool management, including dependency conflicts, version mismatches, and compatibility issues. These challenges impact eSim's usability and efficiency, reinforcing the need for an automated solution.

The theoretical background explores eSim's reliance on Ngspice and KiCad, reviews existing tool management solutions, and identifies their limitations. Additionally, it outlines key technologies like Python, package managers, and system configuration strategies that support automation.

A detailed explanation of the system's design and implementation follows, describing its standalone architecture with modules for installation, updates, and configuration management. Features such as real-time dependency checking and automated updates are discussed, along with challenges like cross-platform compatibility and their solutions. The chapter concludes with testing and validation to demonstrate the tool's effectiveness.

Integration into eSim is then examined, specifically for Ubuntu environments. This includes the development of communication protocols, synchronization mechanisms, and a graphical user interface (GUI) for seamless management. Extensive testing confirms improvements in performance, stability, and user experience.

The final section summarizes the Tool Manager's impact, highlighting benefits while acknowledging limitations such as separate workflows, lack of Windows support, and outdated version display on Ubuntu. Future improvements focus on merging installation and updates, enhancing user experience, expanding Windows compatibility, and incorporating additional simulation tools like LTspice and Qucs.

Chapter 2

Problem Statement

The goal of this task is to design an "Automated Tool Manager" for eSim(preferably in python) that automates the installation, configuration, update, and management of external tools and dependencies.

The requirements for the system are as follows:

1. **Tool Installation Management:**
    - The manager should be able to download and install required external tools (e.g., Ngspice, KiCad, etc.) automatically.
    - Ensure compatibility with the system environment (Linux/Windows).
    - Handle version control to ensure the correct versions of tools are installed.
2. **Update and Upgrade System:**
    - Design a system that checks for updates for the external tools and libraries.
    - Implement functionality for the user to update these tools with minimal manual intervention.
3. **Configuration Handling:**
    - Automate the configuration of the installed tools based on user settings or predefined settings.
    - Handle path management and environment variable settings so that the tools work seamlessly with eSim.
4. **Dependency Checker:**
    - Ensure that all necessary dependencies for the install d tools are checked and managed.
    - Provide feedback to the user if certain dependencies are missing or incompatible.
5. **User Interface:**

- Provide a simple, user-friendly command-line or graphical interface for managing the tools.
- Allow the user to view installed tools, versions, and any updates available.
- Provide a log of actions taken (e.g., installations, updates, errors).

6. **Additional Features (Optional):**
   - Cross-platform support (Linux/Windows/Mac).
   - Integration with popular package managers (e.g., apt, Homebrew, Chocolatey) to streamline tool management.

Chapter 3

Methodology

Developing an Automated Tool Manager for eSim requires expertise in electronic circuit simulation, Python programming, Linux system administration, and NSIS scripting. Knowledge of tools like Ngspice, KiCad, LLVM, GHDL, Verilator, Makerchip, and SkyWater SKY130 PDK is essential for ensuring compatibility and seamless integration with eSim. Research skills in tool dependency management are crucial to address version control and compatibility issues. Additionally, Python package management and automation are key to simplifying installations and updates, forming the foundation for an efficient and reliable tool manager.

3.1 Operating Systems

Windows 11, released on October 5, 2021, offers a modern UI, enhanced security, and improved performance. With Chocolatey for package management, it supports seamless tool automation in the Automated Tool Manager for eSim, ensuring compatibility with Ngspice, KiCad, LLVM, and GHDL.

Ubuntu 20.04 LTS (Focal Fossa), released on April 23, 2020, is a stable Linux distribution with long-term support until 2025. Featuring the 5.4 Linux kernel, GNOME 3.36, and APT/Snap package management, it provides a reliable environment for eSim tools like Ngspice, KiCad, LLVM, and GHDL.

3.2 VirtualBox

VirtualBox, developed by Oracle, is an open-source virtualization tool for running multiple operating systems on a single machine. It enables isolated test environments, supports various guest OSes, advanced networking, and snapshots, making it ideal for the Automated Tool Manager for eSim. VirtualBox allows seamless testing of tools like Ngspice, KiCad, LLVM, and GHDL across platforms without affecting the host system.

3.3 eSim

eSim, developed by the FOSSEE project at IIT Bombay, is an open-source EDA tool for circuit design, simulation, and PCB layout. It integrates Ngspice for simulation and KiCad for PCB design, supporting Windows 11 and Ubuntu 20.04. As a cost-effective

alternative to proprietary tools, eSim enables transient analysis, frequency response, and noise evaluation, driving innovation in electronics and related fields.

3.4 Python

Python is ideal for developing the Automated Tool Manager for eSim due to its simplicity, rich libraries, and cross-platform support. It automates tool installation, configuration, and dependency management with modules like os and subprocess, while pip and virtual environments prevent conflicts. Libraries like PyQt and Tkinter enable intuitive interfaces, enhancing usability. Python's modularity ensures maintainability, streamlining workflows and reducing manual effort in managing eSim tools.

3.5 Linux system administration

Linux system administration is crucial for developing the Automated Tool Manager for eSim, ensuring efficient management of tools and dependencies. Proficiency in Linux enables package installation, permission management, environment configuration, and shell profile editing for seamless tool integration. Key skills include troubleshooting with system logs and diagnostic commands to maintain tools like Ngspice, KiCad, and GHDL. Linux's stability and command-line utilities support automation of dependency resolution, version control, and configuration, ensuring a reliable platform for eSim users.

3.6 NSIS

NSIS (Nullsoft Scriptable Install System) is essential for creating Windows installers in the Automated Tool Manager for eSim, enabling seamless packaging and distribution of tools and dependencies. Its scripting capabilities allow file extraction, registry modifications, and environment configuration for tools like Ngspice, KiCad, and GHDL. NSIS supports conditional logic for adaptive installations, prerequisite checks, and version management. Its lightweight design, broad Windows compatibility, and user-friendly features ensure efficient tool installation and configuration, enhancing usability for eSim users.

3.7 Research Skills in Tool Compatibility and Dependency Management

Research skills in tool compatibility and dependency management are vital for developing the Automated Tool Manager for eSim. Managing tools like Ngspice, KiCad, LLVM, GHDL, Verilator, and Makerchip requires understanding their requirements, versions, and interdependencies across Linux and Windows. Key tasks include analyzing OS requirements, tracking dependencies, resolving conflicts, and handling deprecated libraries. Effective research leverages documentation, forums, and repositories for troubleshooting. By automating dependency checks, version control, and compatibility validation, the Tool Manager ensures a seamless and adaptable workflow for eSim users. The tools and packages use in eSim are as follows:

- Ngspice – An open-source SPICE-based circuit simulator for transient, AC, and DC analysis, integrated with eSim for analog and mixed-signal simulations.
- KiCad – An open-source PCB design tool supporting schematic capture, multi-layer PCB layout, component libraries, and 3D modeling.
- LLVM – A modular compiler and toolchain framework used in hardware description languages (HDL) and high-performance electronic design.
- GHDL – An open-source VHDL simulator for analysis, synthesis, and verification of digital circuit designs.
- Verilator – A high-speed, open-source Verilog simulator that converts Verilog to C++ or SystemC for efficient simulation.
- Makerchip – An online platform for developing and simulating digital circuits using Verilog and TL-Verilog with an intuitive interface.
- SkyWater SKY130 PDK – An open-source process design kit for IC design, providing libraries, design rules, and SKY130 technology support.
- PyQt5 – A Python binding for Qt, enabling cross-platform GUI development for eSim User Interface.

3.8 Chocolatey

Chocolatey is a powerful Windows package manager that automates software installation, management, and configuration, making it ideal for the Automated Tool Manager for eSim. It simplifies dependency management with single-command installations of tools like Ngspice, KiCad, and LLVM while supporting version control to minimize compatibility issues. Its scriptable, silent installations and integration with PowerShell and CI/CD pipelines enhance automation. By leveraging Chocolatey, the Tool Manager ensures consistent installations, reduces manual effort, and improves reliability in managing eSim's tools and dependencies.

3.9 APT (Advanced Package Tool)

APT (Advanced Package Tool) is a package manager for Debian-based Linux systems like Ubuntu, crucial for the Automated Tool Manager for eSim. It automates installing, updating, and managing tools like Ngspice, KiCad, and GHDL while resolving dependencies and ensuring compatibility. APT fetches updates, removes outdated packages, and integrates seamlessly into workflows for efficient tool management. Its automation minimizes manual effort, ensuring a reliable Linux environment for eSim.

Chapter 4

System Implementation (Standalone App)

The Automated Tool Manager streamlines external tool management, automating the installation of Ngspice, KiCad, and others while ensuring compatibility across Linux and Windows. It handles version control, updates, and dependency resolution for seamless integration. Developed with PyQt5 or as a CLI tool, it provides an intuitive interface for managing tools, viewing versions, and accessing logs. Integration with package managers like APT and Chocolatey enhances efficiency, reducing manual effort for eSim users.

4.1 System Methodology

The Automated Tool Manager simplifies tool and dependency management for seamless eSim integration. It automates installation, detects the system environment (Linux/Windows), and ensures compatibility using package managers like APT and Chocolatey. A built-in version control mechanism installs the correct tool versions, while automated update checks notify users and facilitate upgrades with minimal effort. Configuration management handles environment variables and system paths, ensuring proper tool integration. A dependency checker detects and resolves conflicts, providing logs and alerts for troubleshooting.

With a Python-based interface (PyQt5 or CLI), users can easily manage tools, check versions, and monitor logs. The system supports cross-platform functionality, leveraging package managers like Homebrew, APT, and Chocolatey, and is designed to be scalable for future requirements.

4.2 Overall System Design

The Automated Tool Manager features a modular architecture for scalability, maintainability, and efficient tool management. It initializes by setting up the environment, with a central module managing user interactions and delegating tasks to dedicated modules. Modules handle installation, updates, and configuration for Chocolatey, Ngspice, KiCad, LLVM, GHDL, and Python packages. The Chocolatey module automates Windows installations, while other modules ensure seamless integration with eSim. A Python package manager handles PyQt5, virtual environments, and dependencies.

The system automates installation, updates, and dependency checks, reducing user intervention. Supporting Linux, Windows, and potentially macOS, it offers a CLI or graphical interface for managing tools, viewing versions, and accessing logs. This modular, automated design enhances usability and allows future expansion.

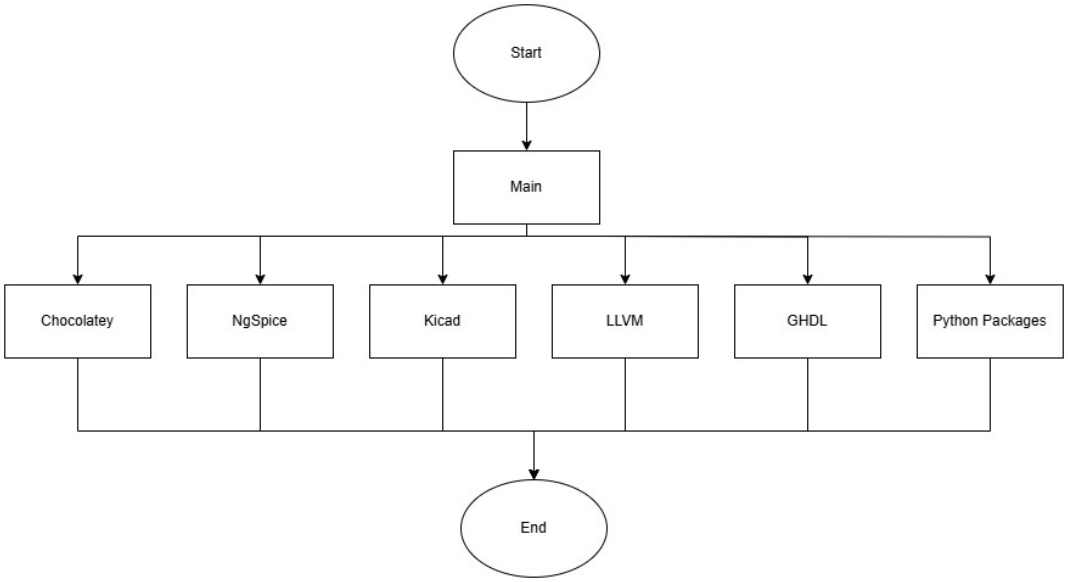

Figure 4.1: Overall System Design

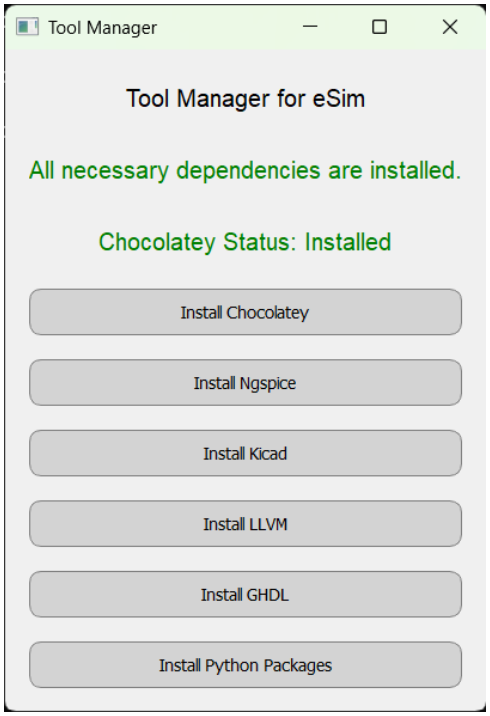

Figure 4.2: eSim Tool Manager for Windows

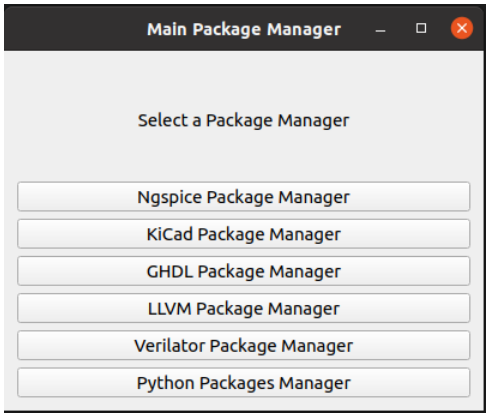

Figure 4.3: eSim Tool Manager for Ubuntu

4.3 Chocolatey Workflow (Only for Windows)

The workflow for managing packages with Chocolatey in the Automated Tool Manager begins with the user choosing an action: installation or update. For installation, the system checks if the package is already installed. If not, it installs the package; otherwise, it informs the user to avoid redundancy. For updates, the system verifies if the package is up-to-date. If not, it updates the package; otherwise, it notifies the user they are on the latest version.

This efficient process automates status and version checks, minimizes unnecessary actions, provides clear user feedback, and ensures an up-to-date environment, enhancing the Tool Manager's reliability and usability.

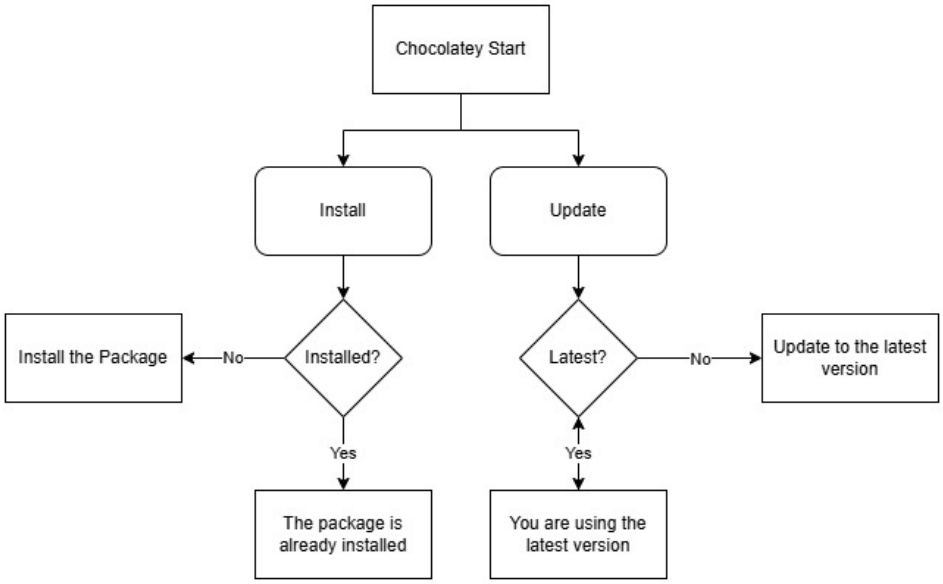

Figure 4.4: Chocolatey Workflow

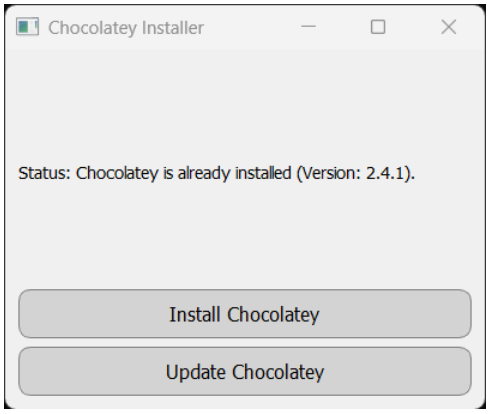

Figure 4.5: Chocolatey GUI

4.4 Ngspice Workflow

The Automated Tool Manager streamlines Ngspice management by allowing users to select a specific version through the Ngspice Start module, ensuring compatibility with their projects or access to the latest features. Users can choose to install or update Ngspice. During installation, the system checks if the selected version is already installed; if not, it proceeds with installation and resolves dependencies, otherwise notifying the user to prevent redundancy. For updates, the system verifies if the latest version is installed and updates if necessary; otherwise, it informs the user that no action is required. This automated workflow simplifies version control, minimizes manual intervention, and ensures users always have the appropriate Ngspice version for their needs.

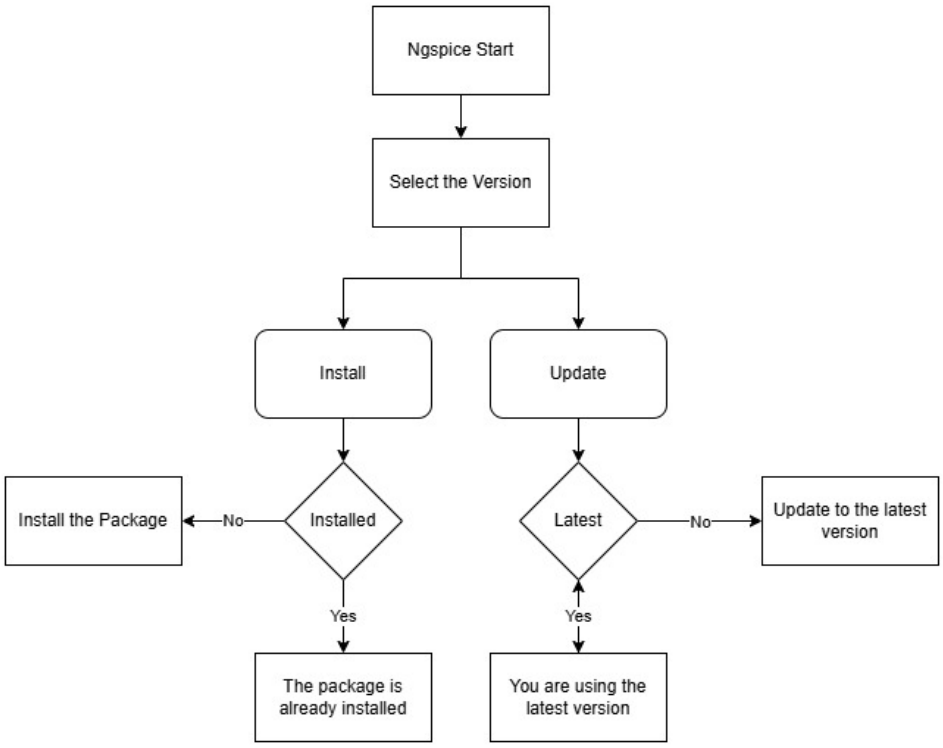

Figure 4.6: Ngspice Workflow

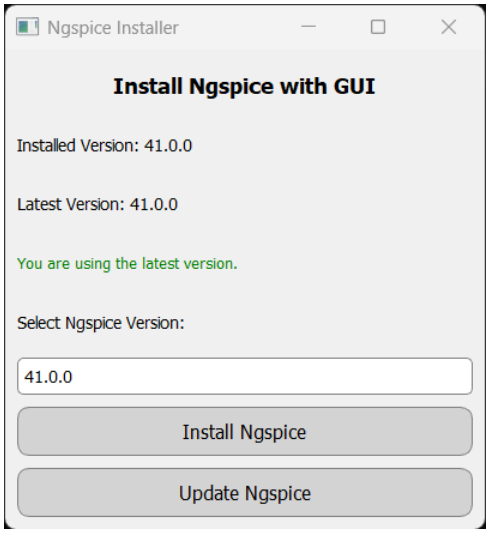

Figure 4.7: Ngspice GUI for Windows

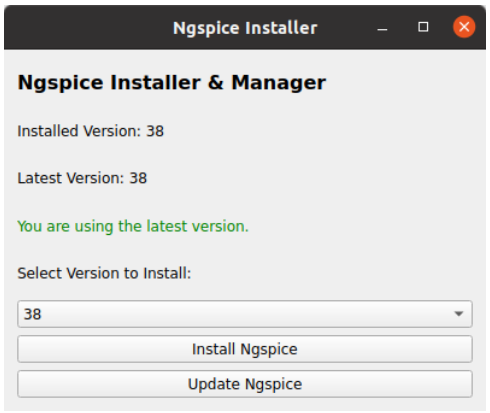

Figure 4.8: Ngspice GUI for ubuntu

4.5 Kicad Workflow

The workflow for managing KiCad in the Automated Tool Manager begins with the KiCad Start module, where the user selects a specific version for installation or update, ensuring flexibility for project compatibility or access to the latest features. For installation, the system checks if the selected version is already installed. If not, it installs the version, resolving all dependencies. If the version is already installed, the user is notified to avoid redundancy. For updates, the system verifies if the installed version matches the latest available. If outdated, it updates KiCad automatically; otherwise, the user is informed that no action is needed.

This streamlined process automates version selection, installation, and updates, reducing manual effort and ensuring compatibility with eSim. Clear feedback at each step ensures a user-friendly experience while keeping KiCad optimized and up-to-date.

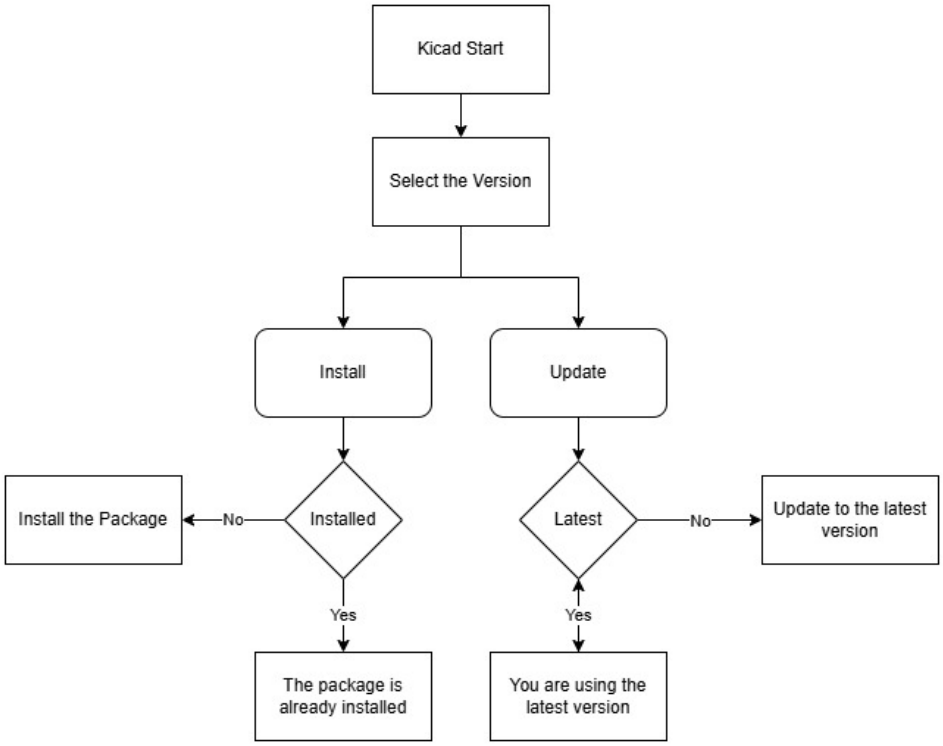

Figure 4.9: Kicad Workflow

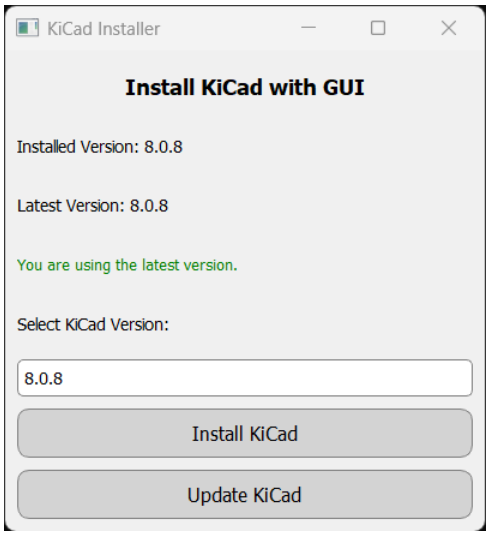

Figure 4.10: Kicad GUI for Windows

Figure 4.11: Kicad GUI for Ubuntu

4.6 LLVM Workflow

The LLVM workflow in the Automated Tool Manager starts with the LLVM Start module, where users select a specific or latest version of LLVM. For installation, the system checks if the version is already installed. If not, it installs the package and resolves dependencies. If installed, the user is notified to prevent redundancy. For updates, the system checks if the version is the latest. If outdated, it updates LLVM; otherwise, it informs the user no action is needed. This streamlined process automates installations and updates, ensures compatibility, and provides clear feedback for efficient LLVM management.

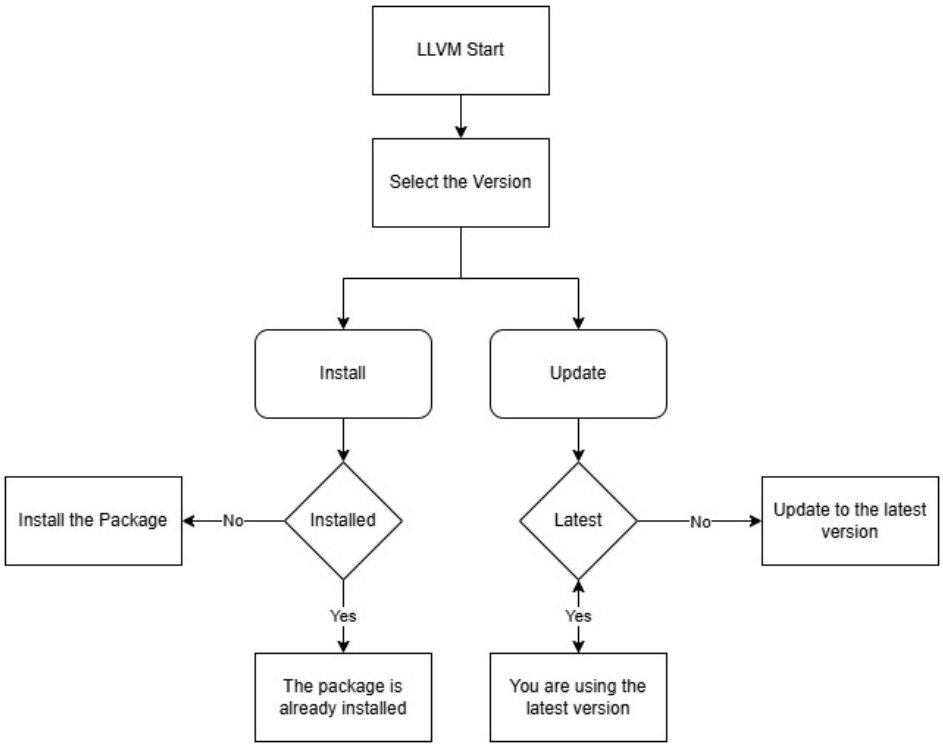

Figure 4.12: LLVM Workflow

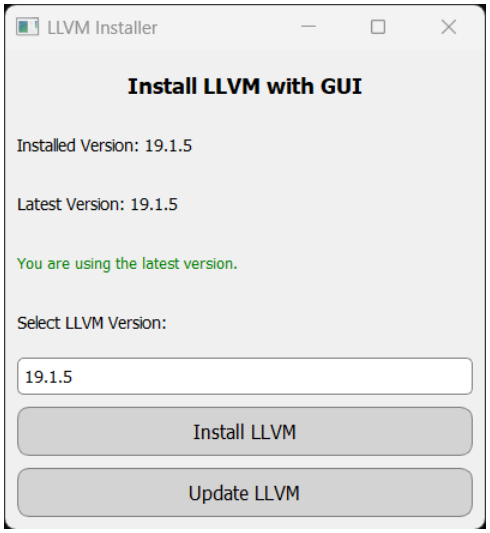

Figure 4.13: LLVM GUI for Windows

Figure 4.14: LLVM GUI for Ubuntu

4.7 GHDL Workflow

The GHDL workflow in the Automated Tool Manager starts with the GHDL Start module, where users choose to install or update. For installation, the system checks if the selected version is already installed. If not, it installs the package and resolves dependencies; otherwise, it notifies the user to avoid redundancy. For updates, the system verifies if the installed version is the latest. If outdated, it updates to the latest version; if up-to-date, it informs the user no action is needed. This automated workflow streamlines GHDL management, reduces user effort, and ensures compatibility and optimal functionality for eSim users.

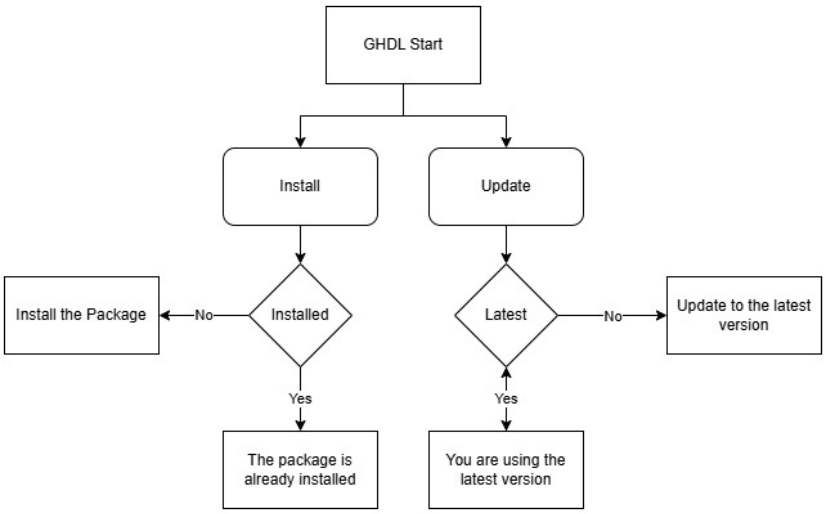

Figure 4.15: GHDL Workflow

Figure 4.16: GHDL GUI for Windows

Figure 4.17: GHDL GUI for Ubuntu

4.8 Python Packages Workflow

The Python Packages workflow in the Automated Tool Manager starts with the Python Package Start module, where users can check installed packages, install new ones, or update existing ones. For Check Installed Packages, the system verifies required packages. If all are present, the user is notified; otherwise, missing packages are identified. For Install, the specified package is installed, resolving dependencies. For Update, the system checks if the package is up-to-date. If outdated, it updates the package; if current, the user is informed. This automated workflow simplifies Python package management, ensuring efficient installation and updates while maintaining a compatible and up-to-date environment.

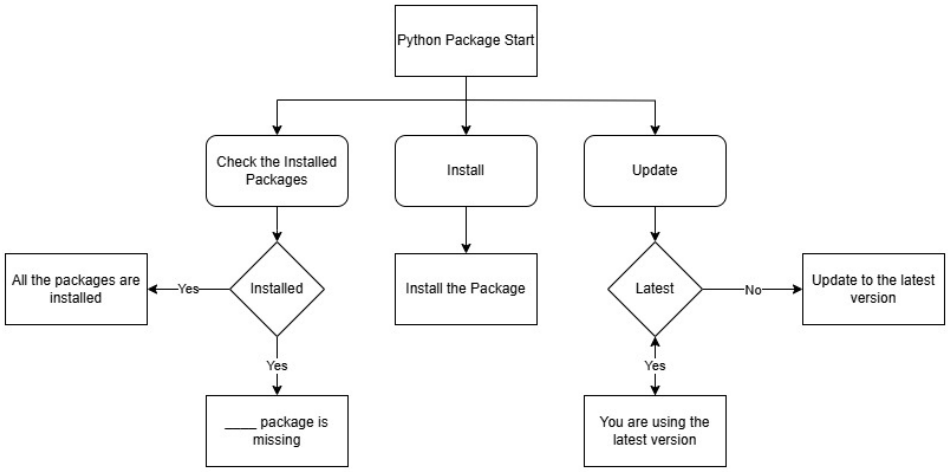

Figure 4.18: Python Packages Workflow

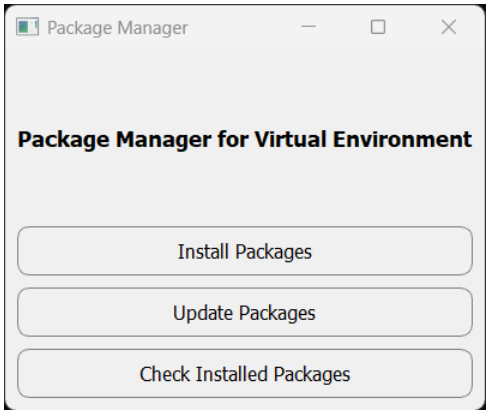

Figure 4.19: Python Packages GUI for Windows

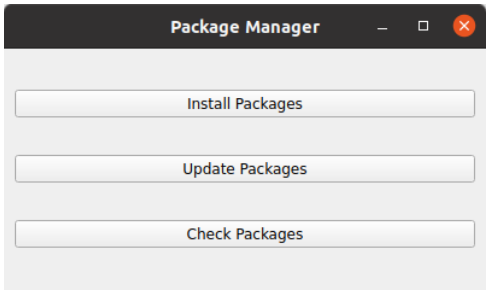

Figure 4.20: Python Packages GUI for Ubuntu

4.9 Testing and Validation Process for User Experience

The testing and validation process was conducted in a systematic and rigorous manner to ensure that the user interface meets the desired functionality and clarity standards. This section outlines the procedures and outcomes associated with the evaluation of various user interactions and system notifications. Installation Notification: Upon successful installation of a package, the system displays a confirmation pop-up with the message: "Package name version number is successfully installed." This message is verified through automated and manual testing to ensure its prompt appearance immediately after installation.

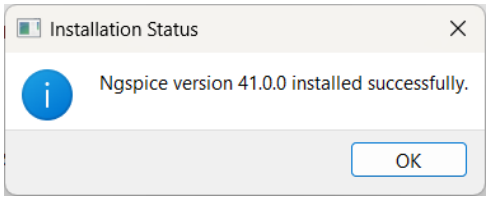

Figure 4.21: Installation Notification (Windows)

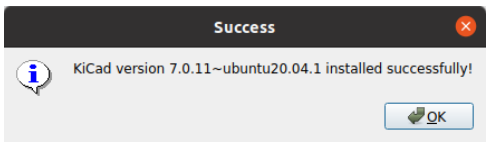

Figure 4.22: Installation Notification (Ubuntu)

Duplicate Installation Attempt: The scenario in which a user attempts to reinstall an already installed package was also tested. In such cases, the system is expected to produce a pop-up notification stating: "The package name is already installed." This validation ensures that the system correctly identifies redundant installation attempts and informs the user accordingly.

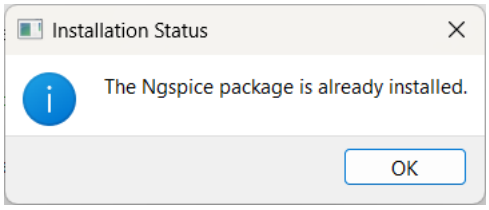

Figure 4.23: Duplicate Installation Attempt (Windows)

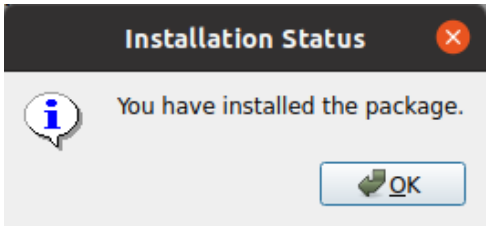

Figure 4.24: Duplicate Installation Attempt (Windows)

Update Notification: In cases where the user initiates an update action while the package is already updated to the latest version, the system is designed to notify the user with the message: "You are using the latest version." Testing of this functionality confirmed that the system reliably detects the current version status and communicates it without error.

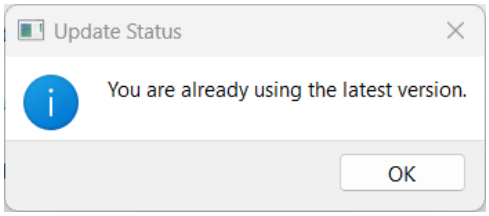

Figure 4.25: Update Notification (Windows)

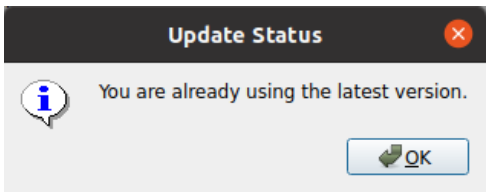

Figure 4.26: Update Notification (Ubuntu)

Post-Installation Confirmation: After closing the installer interface, the system is programmed to display a final confirmation pop-up: "The Package name installation started successfully." This final notification is verified to ensure that the user receives immediate feedback upon completion of the installation process.

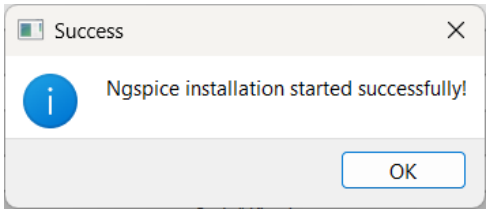

Figure 4.27: Post Installation Confirmation

4.10 Tool Management Features

The Automated Tool Manager includes comprehensive features to manage and track the details of installed packages, ensuring a robust and user-friendly experience. The system maintains detailed records for each package, including the version, installation date, and installation directory, as highlighted in the provided data. This information helps users monitor their tools effectively and ensures compatibility with the eSim environment.

For instance, the Tool Manager tracks essential tools like Ngspice, KiCad, LLVM, and GHDL. It logs their installed status, specific versions (e.g., Ngspice version 41.0.0), and installation paths (e.g., C:\FOSSEE\eSim\Application_Venv\tools\ngspice). Additionally, it manages Python packages, dependencies, and other utilities, ensuring the system is fully equipped for all required tasks.

To ensure transparency and ease of troubleshooting, the Tool Manager incorporates an advanced logging mechanism. All system actions, including installations, updates, errors, and warnings, are recorded in a log file (tool_manager.log). Logs are appended with timestamps and message levels (INFO, WARNING, ERROR) for clarity and traceability. This feature helps users and developers identify and address issues quickly.

These features enhance the reliability of the Tool Manager by keeping users informed about system states, installed tools, and potential issues, ensuring an efficient and seamless management process.

Package	**Version**	**Release Date**
Ngspice	43	July 13, 2024
Kicad	8.0.9	February 19, 2025
LLVM	19.1.5	January 15, 2025
GHDL	v4.1.0	December 19, 2024
Chocolatey	2.4.1	November 23, 2024

Table 4.1: Package Latest Version and Release Date

Chapter 5

System Implementation (Integrated - Ubuntu)

This chapter details the integration of a standalone installer and updater within eSim on Ubuntu. The system automates the downloading and installation of essential tools like Ngspice and KiCad while ensuring compatibility and version control. An update subsystem continuously monitors for tool and library updates, allowing users to upgrade with minimal effort. Automated configuration manages file paths and environment variables for seamless integration with eSim. A built-in dependency checker ensures all required components are present, providing immediate feedback on issues. Additionally, a user-friendly graphical interface enables easy management of installations, updates, and logs.

5.1 System Methodology

The Tool Manager serves as the primary GUI for managing eSim installation, updates, and uninstallation. Users can install eSim Analog Mode (requiring KiCad and Ngspice) or eSim Digital Mode (which also includes GHDL and Verilator). The interface allows for seamless selection, updating, and removal of packages as needed.

The Update Manager provides a clear view of installed versions for KiCad, Ngspice, GHDL, and Verilator, enabling informed updates. Operating independently of Ubuntu versions, it downloads compressed packages to ensure update consistency and avoid dependency issues. After downloading, Python dependencies are updated first, followed by the requested tool versions.

Upon completion, the system generates a JSON file with package details and logs errors or warnings for transparency and troubleshooting. Uninstallation options allow users to remove only digital packages (GHDL and Verilator) while retaining KiCad and Ngspice, or perform a full removal of all eSim-related software. To conserve storage, the system automatically deletes downloaded packages post-update. The user-friendly GUI ensures efficient package management, providing clear insights through logs and JSON records.

5.2 Overall System Design (Tool Manager)

The Tool Manager for eSim provides an intuitive graphical user interface (GUI) for efficiently managing the installation and uninstallation of essential eSim packages. Users can select between Analog Mode, which requires KiCad and Ngspice, and Digital Mode, which additionally includes GHDL and Verilator. The system streamlines the setup process by allowing users to install their preferred mode with a single click, eliminating the need for manual package management.

For uninstallation, the Tool Manager offers two options: Uninstall Digital Packages, which removes only GHDL and Verilator while retaining KiCad and Ngspice for continued analog simulations, and Uninstall All Packages, which completely removes all eSim-related software, ensuring a full system reset if needed. The interface is designed to be user-friendly, with clearly labeled options that facilitate seamless package management. Furthermore, the system is engineered to function independently of the Ubuntu version, ensuring compatibility across different environments and providing a consistent user experience.

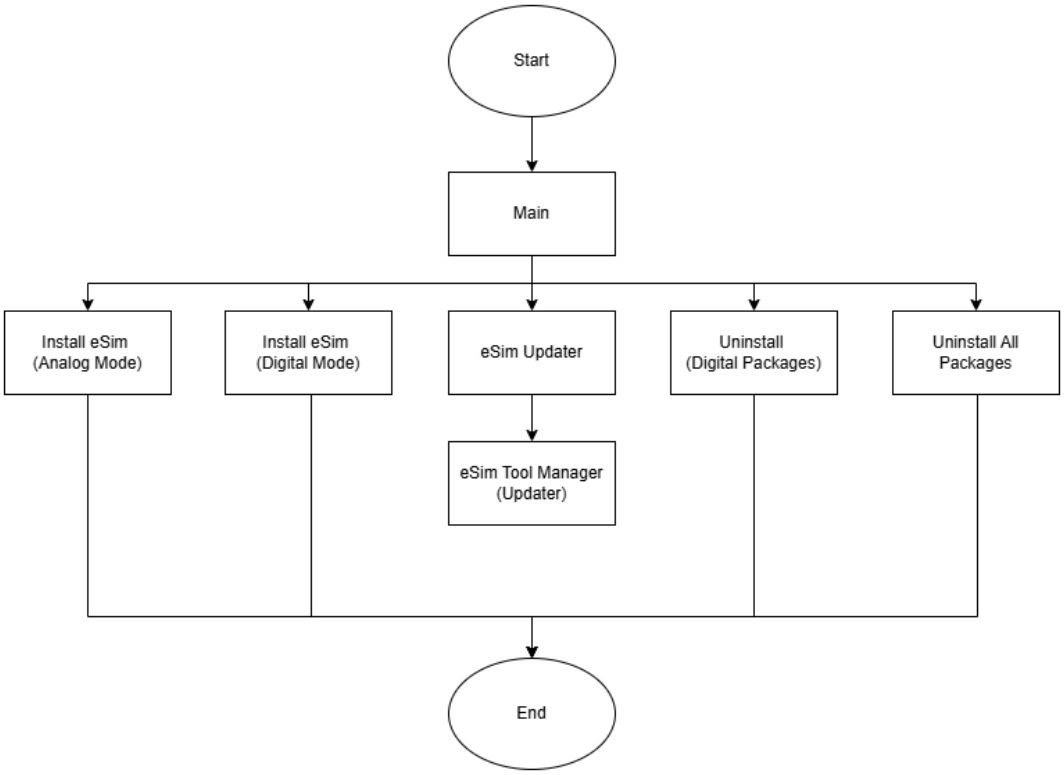

Figure 5.1: Overall System Design (Tool Manager) for Ubuntu (Integrated)

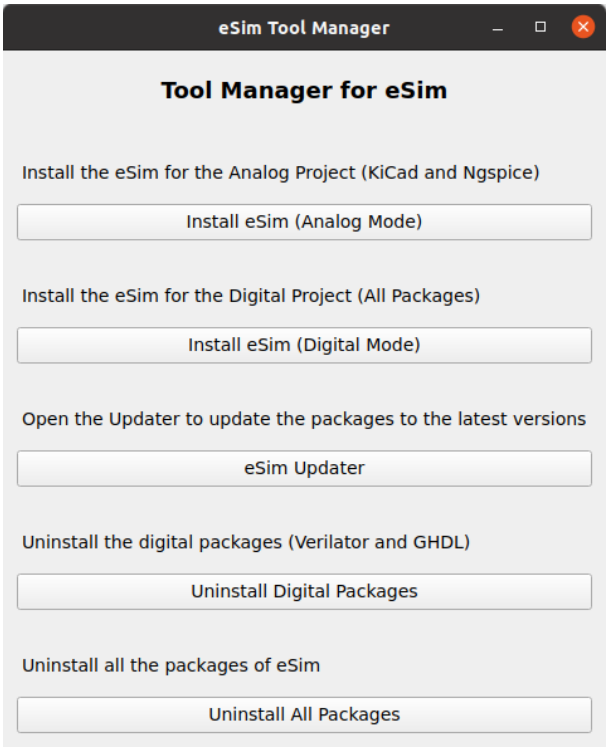

Figure 5.2: Tool Manager GUI for Ubuntu (Integrated)

5.3 Overall System Design (Tool Manager Updater)

The Tool Manager (Updater) system simplifies the installation, updating, and removal of essential tools like KiCad, Ngspice, GHDL, and Verilator through a user-friendly graphical interface (GUI). Users can download missing packages, update dependencies, verify versions, and remove software, ensuring all required components are properly installed and up to date.

Key components include the User Interface (GUI), which displays installed versions and provides controls for downloading, updating, verifying, and removing packages, with status messages guiding users. The Package Download Module ensures necessary tools are installed before updates or version checks. The Dependency Update Module verifies dependencies before applying updates. Each tool has a dedicated updater GUI for individual management. The Package Verification Module compares installed versions against a JSON file, alerting users to discrepancies, while the Package Removal Module allows software deletion when needed. The system follows a structured workflow, prompting users to install missing tools before updates and maintaining accurate version tracking.

A JSON file stores package version data, with the Check Packages function updating it as needed. Clear error messages guide users in resolving issues like missing or outdated dependencies. Overall, the Tool Manager (Updater) system automates software management, reducing manual effort while ensuring compatibility and seamless updates. Its structured workflow enhances usability, making tool management more efficient and reliable.

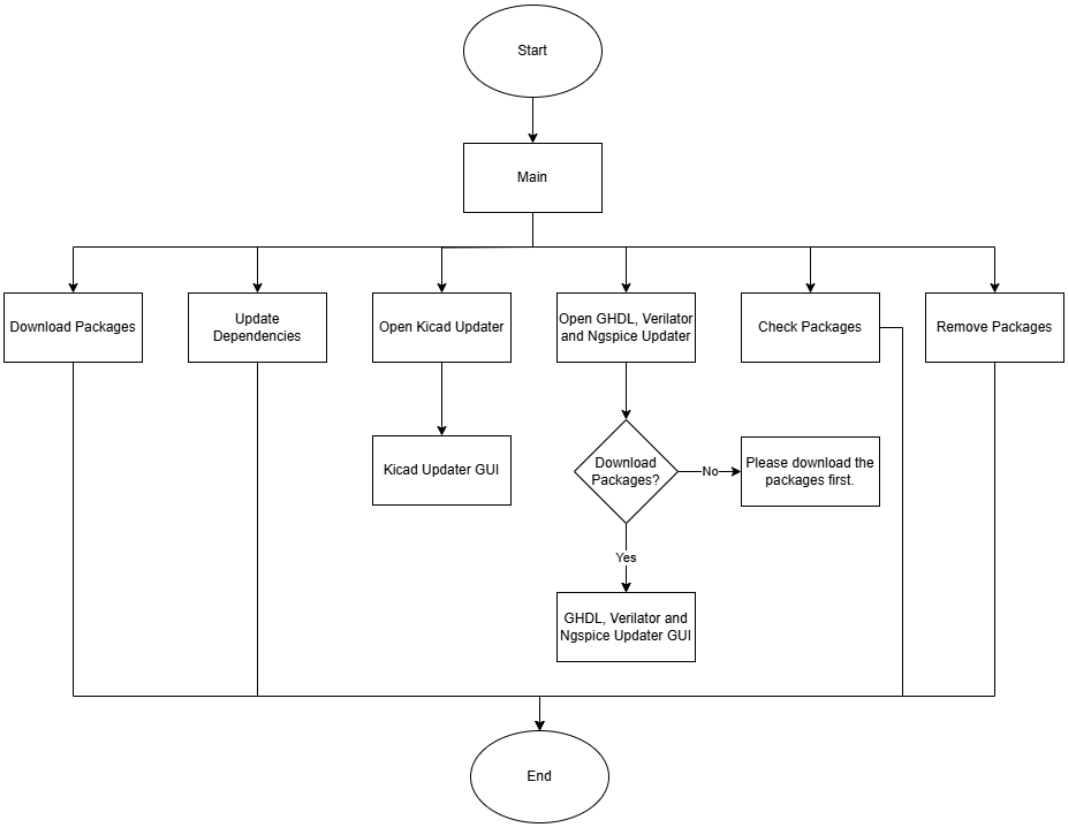

Figure 5.3: Overall System Design of Tool Manager Updater for Ubuntu (Integrated)

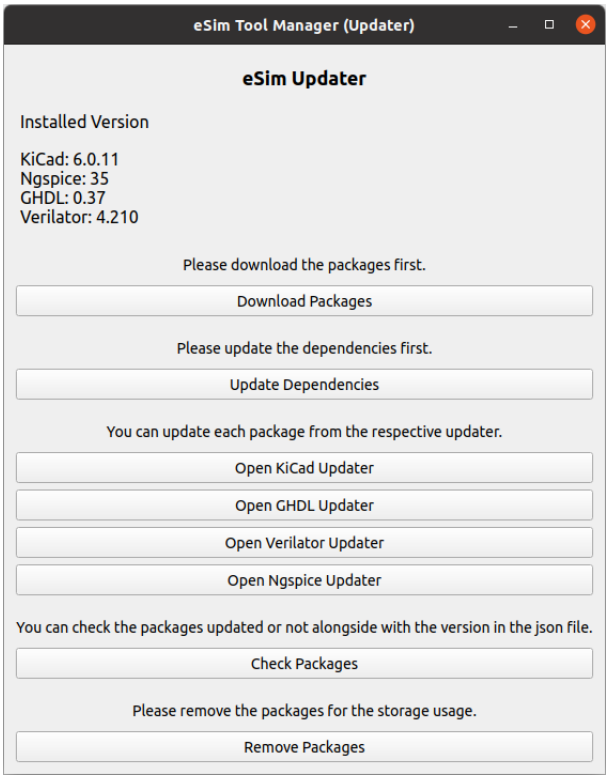

Figure 5.4: Tool Manager Updater GUI for Ubuntu (Integrated)

5.4 Kicad Updater Workflow (Ubuntu)

The KiCad Updater System streamlines the installation and updating of KiCad, ensuring users have the latest version while preserving configurations and libraries. When the KiCad Updater GUI is launched, it automatically checks the installed version. If outdated or missing, a red warning message prompts the user to update. A dropdown menu allows selection of the latest available version, and clicking "Update KiCad" initiates the process.

During the update, the system backs up libraries and symbols, updates KiCad configurations for compatibility, and modifies the JSON file to reflect the new version. Once completed, the GUI reloads, displaying the updated version with a green confirmation message: "You are using the latest version." By automating version detection, configuration management, and updates, the KiCad Updater simplifies software maintenance, ensuring users stay up to date with minimal effort.

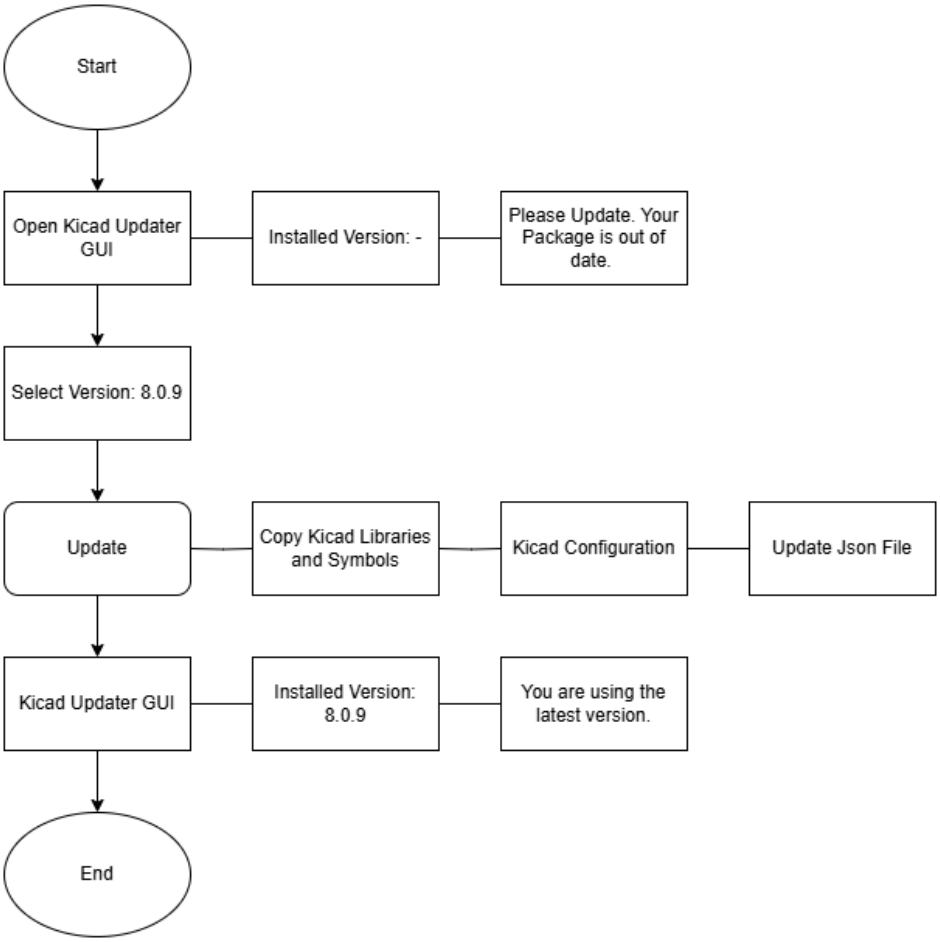

Figure 5.5: Kicad Updater Workflow for Ubuntu (Integrated)

Figure 5.6: Kicad Updater GUI for Ubuntu (Integrated)

5.5 GHDL Updater Workflow (Ubuntu)

The GHDL Updater System simplifies checking, updating, and maintaining the GHDL (VHDL simulator) software. It detects the installed version, prompts for updates, and ensures dependencies are properly configured. When the user opens the GHDL Updater GUI, it checks the installed version. If outdated or missing, a red warning message appears: "Please Update. Your Package is out of date." The user selects the desired version from a dropdown menu and clicks "Update GHDL" to start the update.

The system then updates dependencies, applies GHDL configuration settings, and updates the JSON file to reflect the new version. Once completed, the GHDL Updater GUI refreshes, displaying the latest version with a green confirmation message: "You are using the latest version." This system automates updates, dependency management, and configuration, ensuring a seamless and user-friendly process to keep GHDL up to date.

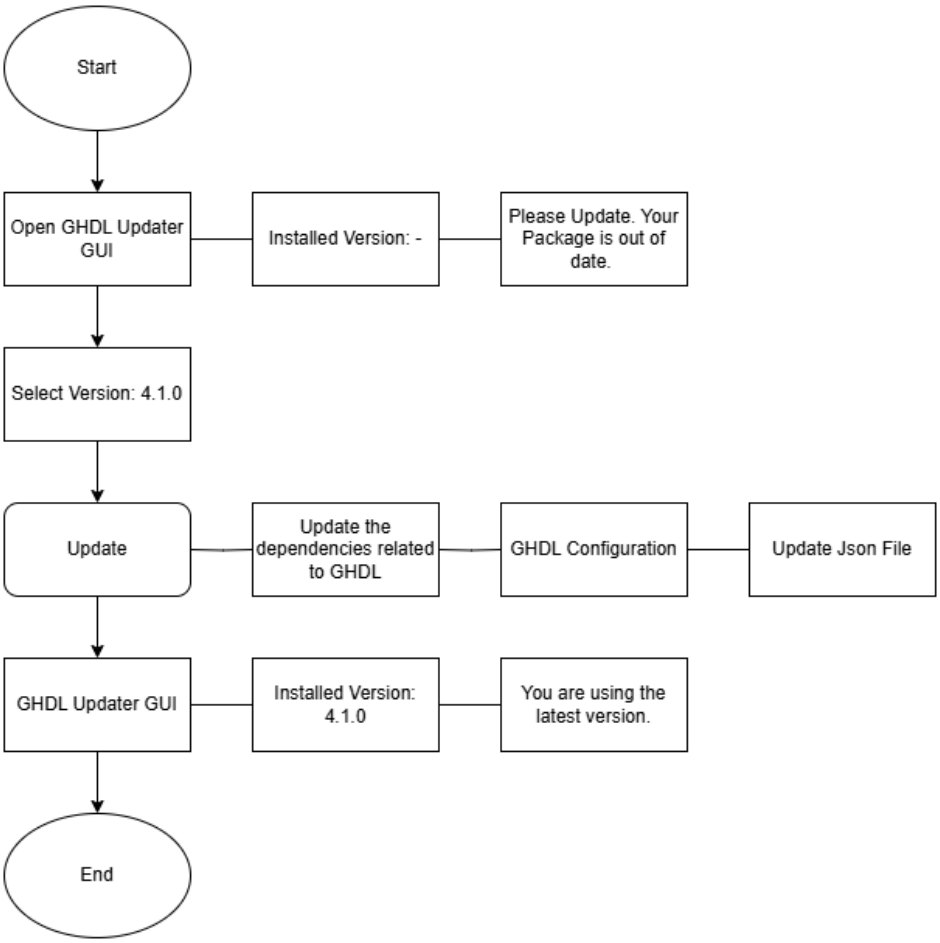

Figure 5.7: GHDL Updater Workflow for Ubuntu (Integrated)

Figure 5.8: GHDL Updater GUI for Ubuntu (Integrated)

5.6 Verilator Updater Workflow (Ubuntu)

The Verilator Updater System streamlines checking, updating, and maintaining the Verilator tool. It detects the installed version, prompts for updates if needed, and ensures dependencies are correctly configured. When the user opens the Verilator Updater GUI, it checks the installed version. If outdated or missing, a red warning message appears: "Please Update. Your Package is out of date." The user selects a version from the dropdown menu and clicks "Update Verilator" to begin the process.

The system then updates dependencies, applies Verilator configuration settings, and updates the JSON file to reflect the new version. Once completed, the Verilator Updater GUI refreshes, displaying the latest version with a green confirmation message: "You are using the latest version." This system automates updates, dependency management, and configuration, ensuring a seamless and user-friendly process to keep Verilator up to date.

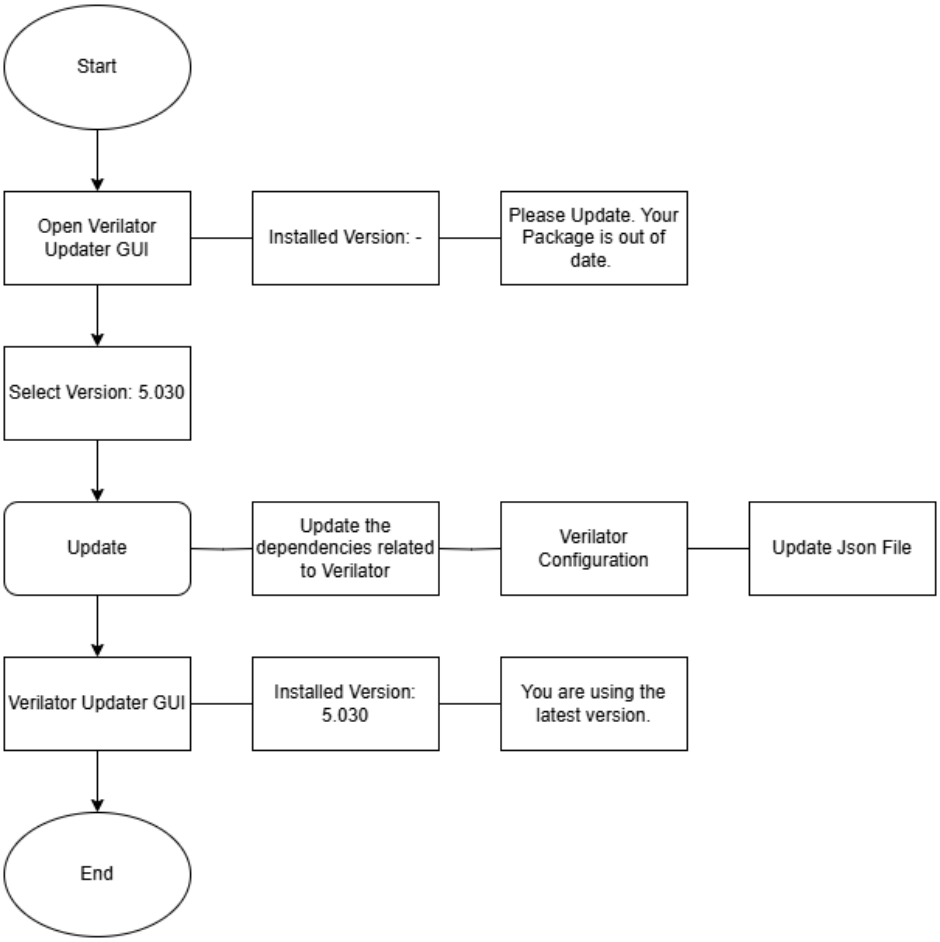

Figure 5.9: Verilator Updater Workflow for Ubuntu (Integrated)

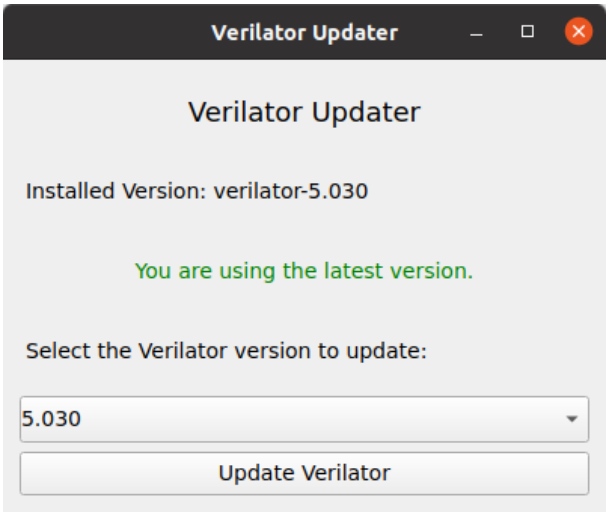

Figure 5.10: Verilator Updater GUI for Ubuntu (Integrated)

5.7 Ngspice Updater Workflow (Ubuntu)

The Ngspice Updater System simplifies checking, updating, and maintaining the Ngspice circuit simulator. It detects the installed version, prompts for updates if needed, and ensures dependencies are properly configured. When the user opens the Ngspice Updater GUI, it checks the installed version. If outdated or missing, a red warning message appears: "Please Update. Your Package is out of date." The user selects a version from the dropdown menu and clicks "Update Ngspice" to begin the process.

The system then updates dependencies, applies Ngspice configuration settings, and updates the JSON file to reflect the new version. Once completed, the Ngspice Updater GUI refreshes, displaying the latest version with a green confirmation message: "You are using the latest version." This system ensures a smooth, automated update process, keeping Ngspice up to date while maintaining an intuitive user experience.

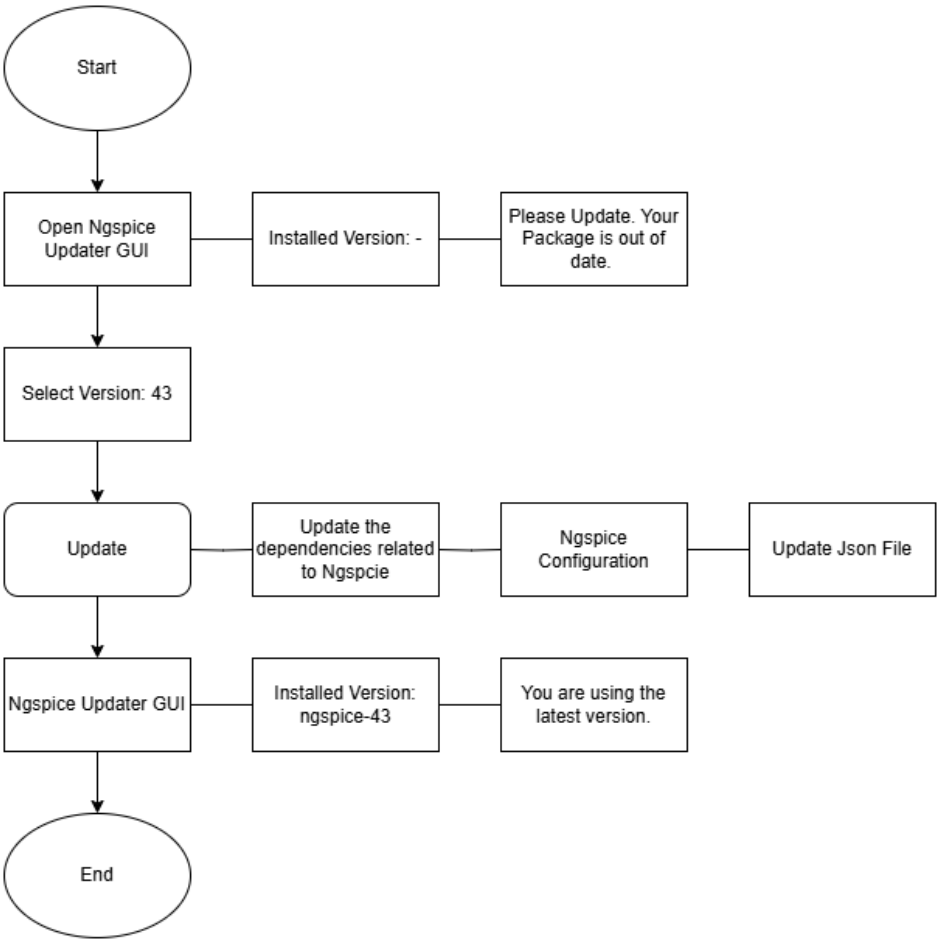

Figure 5.11: Ngspice Updater Workflow for Ubuntu (Integrated)

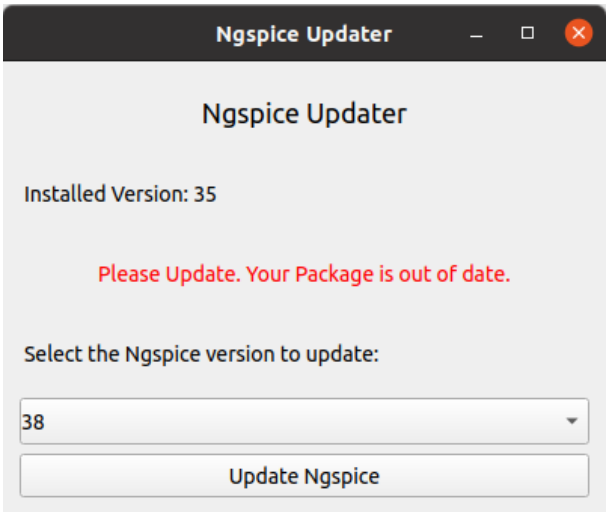

Figure 5.12: Ngspice Updater GUI for Ubuntu (Integrated)

5.8 Testing and Validation Process for User Experience

These system messages provide feedback on various operations within the package manager, ensuring a smooth and user-friendly experience:

Completed Message – "The eSim for analog mode has been successfully installed."

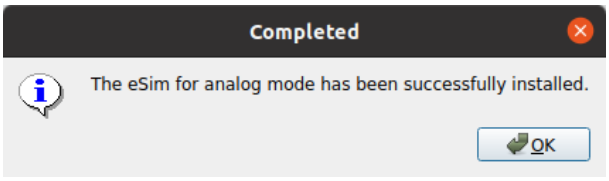

Figure 5.13: Analog Mode installation Completed Message for Ubuntu (Integrated)

This message confirms that the analog packages of eSim have been successfully installed. It assures the user that the installation process has been completed without errors, allowing them to proceed with analog mode simulations.

Completed Message – "The eSim for digital mode has been successfully installed."

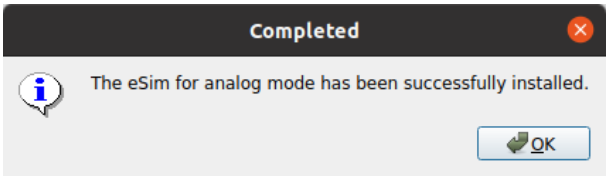

Figure 5.14: Digital Mode installation Completed Message for Ubuntu (Integrated)

This message indicates that the digital packages of eSim have been successfully installed. It provides confirmation that the user can now utilize the digital simulation features without any installation issues.

Completed Message – "The digital packages (Verilator and GHDL) have been successfully uninstalled."

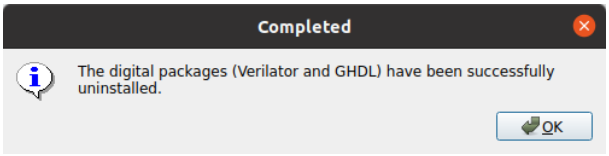

Figure 5.15: Digital Mode uninstallation Completed Message for Ubuntu (Integrated)

This message appears when the user successfully removes the digital packages, including Verilator and GHDL. It ensures that the user is informed about the successful uninstallation, preventing any confusion regarding the status of the digital components.

Completed Message – "All eSim packages have been successfully uninstalled."

Figure 5.16: All Packages uninstallation Completed Message for Ubuntu (Integrated)

This message notifies the user that all eSim-related packages have been completely removed from the system. It serves as a final confirmation that no components remain installed, ensuring a clean uninstallation process.

Error Message – "Please download the packages first."

Figure 5.17: Download Package Error Message for Ubuntu (Integrated)

This indicates that the user attempted to update, check, or remove a package before downloading it. The system prevents further actions until the required packages are installed, ensuring users follow the correct workflow.

Success Message – "Downloaded Packages successfully in the nghdl/packages folder."

Figure 5.18: Downloaded Packages Message for Ubuntu (Integrated)

This confirms that the required packages have been successfully downloaded and stored in the specified directory. It ensures users that the system is ready for updates or installations.

Success Message – "Removed Packages successfully from the nghdl/packages folder."

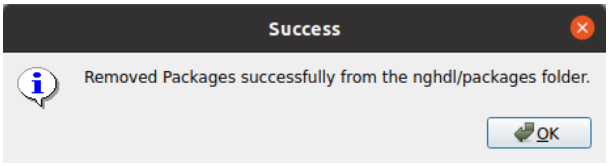

Figure 5.19: Removed Packages Message for Ubuntu (Integrated)

This indicates that selected packages were successfully removed, freeing up storage space. It helps users track and manage package installations efficiently.

Success Message – "KiCad 8.0.9 installation started."

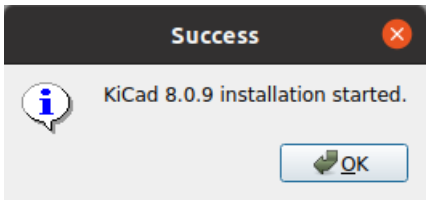

Figure 5.20: Installation Started Message for Ubuntu (Integrated)

This informs the user that the installation of KiCad version 8.0.9 has begun, preventing confusion by providing immediate feedback on installation progress.

Success Message – "KiCad version 7.0.11 ubuntu20.04.1 installed successfully!"

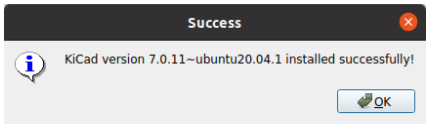

Figure 5.21: Installed Successfully Message for Ubuntu (Integrated)

This confirms that KiCad version 7.0.11 has been installed successfully, ensuring the update process was completed without issues.

Update Status Message – "You are already using the latest version."

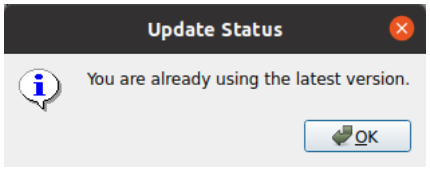

Figure 5.22: Using Latest Version Meassage for Ubuntu (Integrated)

This informs the user that no update is required, as the installed version is up to date. It prevents unnecessary installations and reassures users that their software is current.

These messages enhance user experience by providing clear, immediate feedback on errors and successful operations. By guiding users through each step and validating actions, the system ensures efficient navigation of the package management process with minimal confusion.

Chapter 6

System Implementation (Integrated - Ubuntu)

This chapter discusses the integration of the updater subsystem into eSim on Windows. Unlike the Ubuntu version, which includes both an installer and updater, the Windows version focuses on updating essential external tools like KiCad, Ngspice, GHDL, and Verilator. The updater allows developers to easily add newer versions of these tools for installation. Users can quickly install the latest versions via the updater interface, which simplifies tool management within eSim. A log tracks all update actions for troubleshooting. The updater is equipped with a user-friendly interface that displays the current versions of the installed tools, highlights available updates, and facilitates their installation with just a few clicks. Additionally, a log is maintained, recording all actions taken by the updater for troubleshooting and auditing purposes.

The chapter also describes the automated handling of tool configurations after updates. The updater ensures that all necessary paths and environment variables are properly adjusted to maintain smooth functionality with eSim. Furthermore, the system performs a quick check to ensure that all dependencies are met after an update, providing feedback to the user in case any issues arise.

6.1 Overall System Design (Tool Manager Updater) (Windows)

The Tool Manager (Updater) system is designed to manage the updating process for essential software tools, including KiCad, Ngspice, GHDL, and Verilator. It features a graphical user interface (GUI) that allows users to easily update these tools by opening dedicated updaters for each one. The system ensures that all required tools and dependencies are installed before updates are applied. The workflow begins when the user launches the application and is presented with options to open the tool updaters. Users can download missing packages if necessary, and the system verifies the installed versions against a reference file to notify users of outdated or missing components. The system also includes a package removal option for unnecessary tools.

Core components include the Package Download Module, Dependency Update Module, individual Updater GUIs for each tool, Package Verification Module, and the Package

Removal Module. The system tracks installed versions via a JSON file, ensuring accurate version control and smooth updates. The overall design ensures efficient tool management, guiding users through necessary updates while maintaining system integrity.

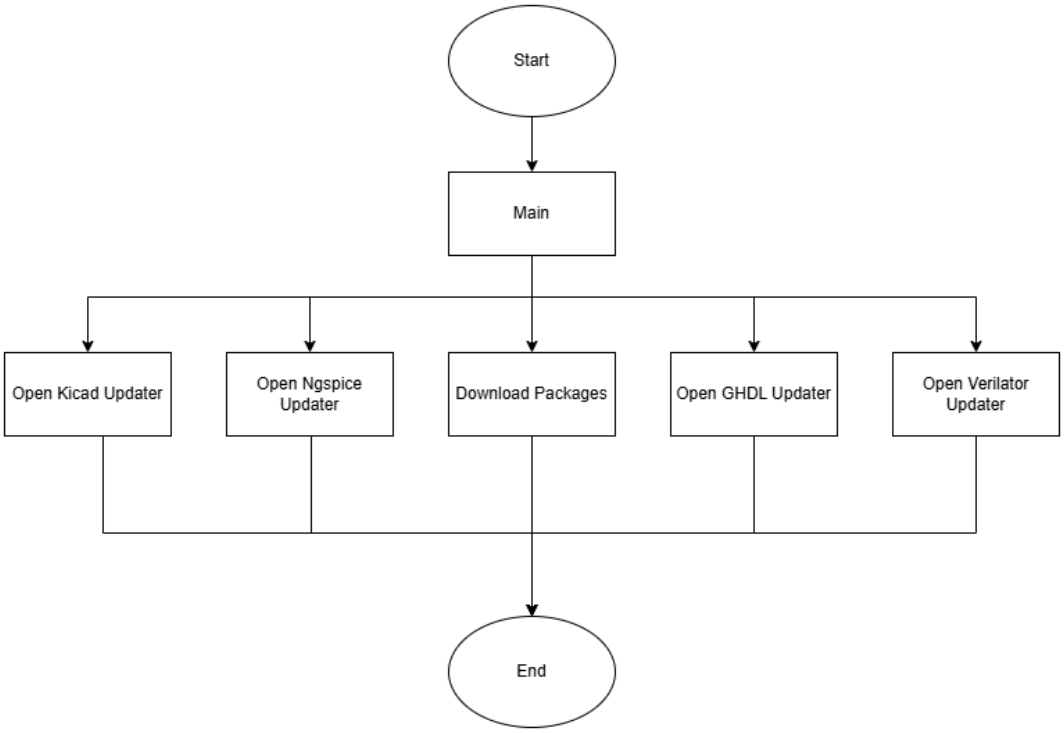

Figure 6.1: Overall System Design of Tool Manager Updater for Windows (Integrated)

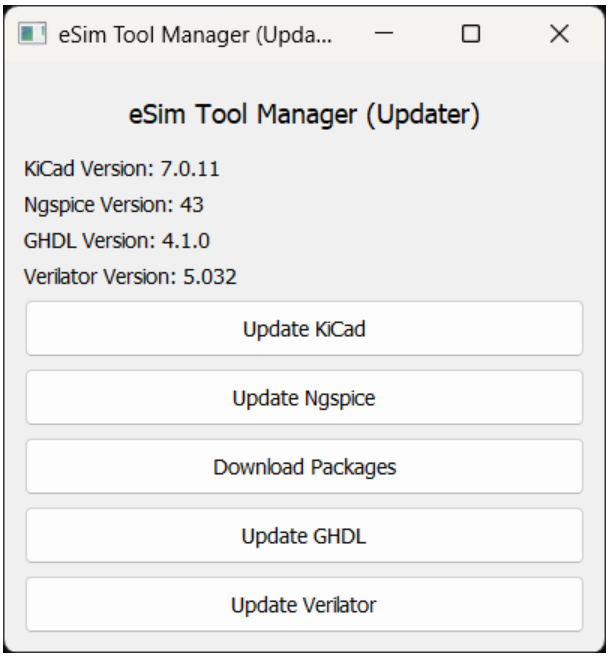

Figure 6.2: Tool Manager Updater GUI for Windows (Integrated)

6.2 Kicad Updater Workflow (Windows)

The KiCad Updater System on Windows ensures that users always have the latest version of KiCad installed. Upon launching the KiCad Updater GUI, the system checks the installed version of KiCad. If the current version is missing or outdated, a message will appear prompting the user to update their package. The user is then presented with a dropdown menu to select the available version of KiCad. Once the desired version is selected, clicking the "Update KiCad" button begins the update process.

During the update, the system copies necessary KiCad libraries and symbols to preserve user configurations and files. Following this, the KiCad configuration is updated to match the new version, ensuring compatibility. Finally, the system updates the JSON file that tracks the installed version, ensuring accurate version control. After the update is completed, the KiCad Updater GUI reloads, displaying the newly installed version and a confirmation message indicating that the latest version is in use. The user can verify that the system is up-to-date with the newly installed version shown in the interface.

This workflow is the same as the Ubuntu version, with the only difference being the platform-specific configurations and installation methods. The Windows version of the KiCad Updater ensures a seamless update process while maintaining user data and settings.

Figure 6.3: Kicad Updater GUI for Windows (Integrated)

6.3 GHDL Updater Workflow (Windows)

The GHDL Updater ensures the GHDL (VHDL simulator) software is up-to-date on Windows. Upon opening the GHDL Updater GUI, the system checks the installed version. If outdated or missing, a warning message appears: "Please Update. Your Package is out of date." The user selects a version from the dropdown menu and clicks "Update GHDL" to begin. The system updates GHDL's dependencies, applies the necessary configuration, and updates the JSON file to reflect the new version. After completion, the GUI reloads with a green message: "You are using the latest version."

The workflow for Windows is the same as the Ubuntu version, with platform-specific

installation and configuration differences. It ensures GHDL is always up-to-date with minimal user intervention.

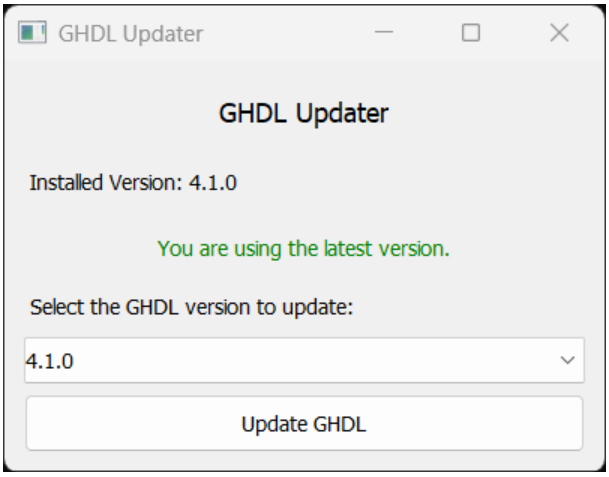

Figure 6.4: GHDL Updater GUI for Windows (Integrated)

6.4 Verilator Updater Workflow (Windows)

The Verilator Updater ensures that the Verilator tool is kept up-to-date on Windows. When the user opens the Verilator Updater GUI, the system checks the installed version. If the version is outdated or missing, a warning message will display: "Please Update. Your Package is out of date." The user selects the version to update from the dropdown menu and clicks "Update Verilator" to begin. The system then updates the necessary dependencies, applies the required Verilator configuration, and updates the JSON file to reflect the new installed version. Once the update is complete, the GUI reloads and displays the latest version with a confirmation message: "You are using the latest version."

The workflow for Verilator on Windows is the same as the Ubuntu version, with platform-specific installation and configuration adjustments. The process ensures that Verilator is always up-to-date and properly configured with minimal user intervention.

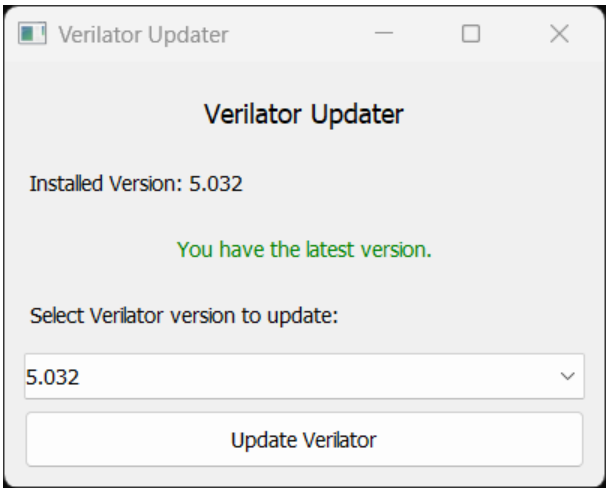

Figure 6.5: Verilator Updater GUI for Windows (Integrated)

6.5 Ngspice Updater Workflow (Windows)

The Ngspice Updater keeps the Ngspice circuit simulator up-to-date on Windows. Upon opening the GUI, the system checks the installed version. If outdated or missing, a warning message appears: "Please Update. Your Package is out of date." The user selects a version from the dropdown and clicks "Update Ngspice" to begin. The system updates dependencies, applies the required configuration, and updates the JSON file to reflect the new version. After completion, the GUI refreshes with a green confirmation message: "You are using the latest version."

The workflow is the same as the Ubuntu version, with platform-specific adjustments. It ensures Ngspice stays updated with minimal user intervention.

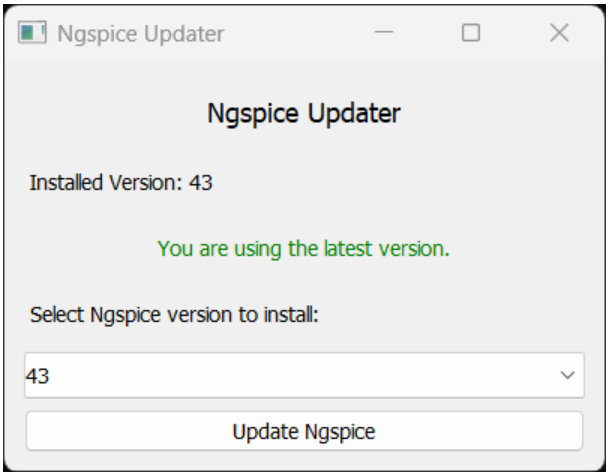

Figure 6.6: Ngspice Updater GUI for Windows (Integrated)

Chapter 7

Conclusion

The Tool Installation and Management System simplifies the installation, updating, and management of external tools like Ngspice and KiCad, ensuring compatibility across Linux and Windows while maintaining version control for consistency. Its update and upgrade mechanism allows users to check for and apply updates effortlessly. Configuration handling automates tool setup by managing paths and environment variables, ensuring seamless eSim integration. A built-in dependency checker verifies and resolves missing or incompatible components, alerting users as needed.

With a user-friendly interface, users can easily view installed tools, check for updates, and log system actions. This system enhances usability, reduces manual effort, and ensures efficient management of external software tools.

7.1 Key Benefits

The system automates installation and updates, reducing manual effort for tools like Ngspice and KiCad while ensuring version control and compatibility. With cross-platform support for Linux and Windows, it manages environment variables and paths, ensuring seamless integration with eSim. Configuration handling is automated, minimizing setup errors and improving efficiency. The dependency checker ensures all required components are installed, preventing compatibility issues. A user-friendly interface allows users to view installed tools, check updates, and log system actions easily. By automating repetitive tasks, the system improves user experience and productivity.

7.2 Limitations

The limitations of the Tool Manager are as follows:

- Separate Installation and Update Workflows: Treating installation and updating as separate processes increases complexity. Merging them into a unified workflow would streamline package management and improve usability.
- User Experience Enhancements: While functional, the system's UI needs better navigation and interaction design for a more intuitive and efficient user experience.
- Lack of Windows Support: Currently limited to Linux, the system needs Windows compatibility to expand accessibility and usability.

- Inaccurate Version Display on Ubuntu: The system version appears outdated even when fully updated, causing confusion. Fixing the version-checking mechanism would ensure accurate status updates.

Addressing these issues will enhance efficiency, usability, and cross-platform support, improving the overall user experience.

7.3 Overall Impact

The Tool Installation and Management System significantly improves the efficiency of installing, updating, and managing essential software tools like Ngspice and KiCad. By automating installation, version control, and dependency management, the system reduces manual effort and minimizes errors, allowing users to focus on their core tasks. Its cross-platform potential, though currently limited to Linux, lays the foundation for broader accessibility in the future. While there are limitations in workflow integration, user experience, and Windows support, addressing these challenges will further enhance its usability and effectiveness. Overall, the system provides a streamlined, automated, and user-centric approach to managing external tools, improving productivity and simplifying software maintenance.

7.4 Future Extensions

The Future Extensions for the Tool Manager are as follows:

- Unified Installation and Update Workflow: Future versions should merge installation and updating into a single process, reducing complexity and improving efficiency.
- User Experience Improvements: Enhancing the UI, navigation, real-time progress indicators, and error handling will make the system more intuitive for all users.
- Windows Compatibility: Expanding support to Windows will increase accessibility, requiring adaptations for package installation, dependency management, and configuration.
- Accurate Version Detection on Ubuntu: A better version-checking mechanism will eliminate discrepancies and ensure users see the correct installed versions.
- Support for More Simulation Tools: Adding tools like LTspice, Qucs, and Xyce will expand functionality, allowing users to manage more electronic design software from a single interface.
- Seamless Automation for Installation and Updates: Automating background updates, dependency resolution, and real-time notifications will create a more robust and user-friendly system.

These improvements will make the system more powerful, intuitive, and comprehensive for managing electronic design and simulation tools.

Bibliography

[1] FOSSEE eSim Project. *eSim Resources*. Available at: `https://esim.fossee.in/resources`

[2] Python Software Foundation. *The Python Programming Language*. Available at: `https://www.python.org/`

[3] Vogt, H. (n.d.). *Ngspice, the open source Spice circuit simulator - Intro.*. Available at: `https://ngspice.sourceforge.io/`

[4] KiCad EDA. (n.d.) *KICAd EDA.*. Available at: `https://www.kicad.org/`

[5] Chocolatey *Chocolatey - The package manager for Windows.(n.d.). Chocolatey Software*. Available at: `https://chocolatey.org/`

[6] LLVM *The LLVM Compiler Infrastructure project.(n.d.)*. Available at: `https://llvm.org/`