Conceptual Architecture Report: GNUStep

February 14th, 2025

CISC 326/322
Aatif Mohammad: 22am37@queensu.ca
Madison MacNeil: 20mkm17@queensu.ca
Avery McLean: 21arcm@queensu.ca
Aden Wong: wong.aden@queensu.ca
Greg Costigan: 21gjc10@queensu.ca
Ryan Vuscan: ryan.vuscan@queensu.ca

1. Abstract

GNUstep is an open-source implementation of the NextStep and OpenStep frameworks, providing a foundation for building cross-platform applications in an object-oriented environment. This document explores the architecture of GNUstep, detailing its derivation, design principles, and underlying components. The report explores the historical evolution of GNUstep, tracing its development from its origins in the NextStep and OpenStep frameworks. GNUstep is modeled using modern software development paradigms, both as a layered and object-oriented architecture. Furthermore, we delve into the structural components of GNUstep's architecture, discussing its core libraries and development tools in depth. By examining the functionality of these components, we gain insight into GNUstep's potential as a viable alternative to proprietary frameworks.

2. Introduction and Overview

GNUstep is a free software implementation of Apple's Cocoa framework, designed to provide a robust environment for application development across multiple platforms. It includes two primary libraries:

2.1 GNUstep-base (Foundation Kit)

This provides fundamental data structures, utilities, and system interface functionalities, allowing developers to create portable applications that integrate seamlessly with different operating systems.

2.2 GNUstep-gui (Application Kit)

This delivers graphical user interface (GUI) components that enable the development of sophisticated and responsive desktop applications.

GNUstep was developed with a focus on modularity and extensibility. By adhering to OpenStep specifications, it ensures that applications built using GNUstep maintain compatibility with other object-oriented development environments, such as Cocoa. This framework is widely used in the development of cross-platform software where consistent user experience is a priority.

Additionally, GNUstep is supported by a dedicated community that continuously improves its performance and expands its capabilities. Developers benefit from a well-documented API, extensive libraries, and comprehensive development tools. These elements contribute to a mature ecosystem that fosters innovation and collaboration.

3. Derivation Process

GNUstep's architecture is derived from the OpenStep specification, which was originally developed by NeXT Inc. and later adopted by Apple as the foundation of the macOS Cocoa framework. The key aspects of its derivation process include:

3.1 Historical Context

OpenStep, developed in the early 1990s, introduced a powerful object-oriented programming model that heavily influenced modern UI frameworks. The design choices made in OpenStep laid the groundwork for GNUstep's evolution as an independent yet compatible framework.

3.2 Core Foundation

GNUstep maintains compatibility with OpenStep by implementing fundamental APIs that provide consistent data handling, event management, and UI controls. This ensures that developers familiar with Cocoa can easily transition to GNUstep without significant re-learning.

3.3 Adaptation and Evolution

Over time, GNUstep has integrated modern programming practices and enhanced support for contemporary system architectures. Unlike its predecessors, GNUstep embraces additional portability features, making it compatible with GNU/Linux, Windows, and other Unix-based operating systems. By leveraging Objective-C's dynamic runtime, GNUstep delivers a flexible and highly extensible framework suitable for various applications.

4. Architecture

4.1 Architecture Style

Like many large scale software applications, GNUstep makes use of multiple architectural styles to provide its functionality.

4.1.1 Layered Style

GNUsteps utilizes a layered style to provide different functionalities across the system, dividing it into layers based on OpenStep and Cocoa. The topmost layer is the development tools layer, which contains an environment for interactive, drag and drop UI development with the Gorm tool, allowing developers to design graphical interfaces. Next, the application kit layer handles the creation of UI objects and events during runtime using the GUI library. The backend layer exists to act as a backend to abstract the UI objects created by the GUI library to the appropriate OS making the application portable. Next is the foundation kit layer which interacts with the application kit providing base services and system commands. Finally the runtime layer provides core libraries as well as objective-c runtime.

4.1.2 Object Oriented Style

GNUstep provides an object-oriented framework based on Objective C, given that it makes use of Objective-C, an object oriented programming language. The object-oriented design allows the system to structure the components in a modular way, allowing for greater flexibility and scalability. GNUstep makes use of Obj-C's protocols and categories. This architecture simplifies the development of applications and supports the creation of dynamic and maintainable software in GNUstep.

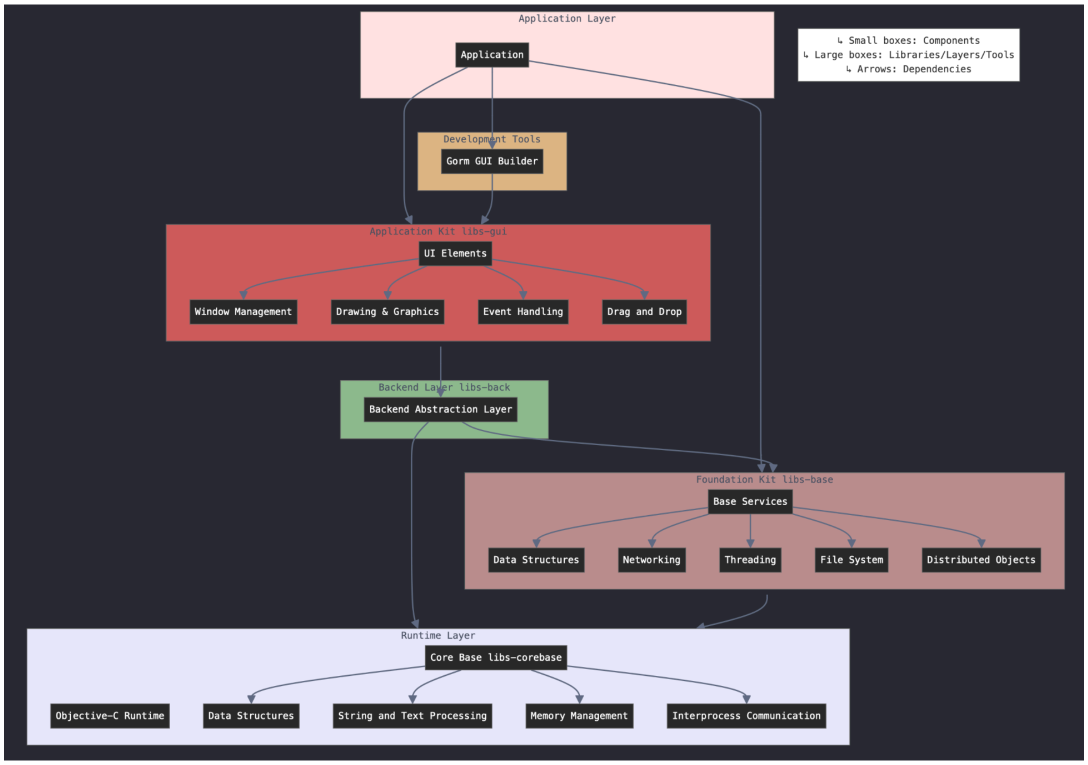


Figure 1: Visualization of GNUstep as a layered software architecture

4.2 System Functionality (Described Top-Down):

4.2.1 Development Tool: Gorm (GNUstep Object Relationship Modeler)

Gorm (Graphical Object Relationship Model) is the GNUstep counterpart to NeXT's Interface Builder and is fully compatible with the Cocoa API, which is used in macOS development. The interface builder allowed for drag and drop placement of elements such as buttons, sliders, text boxes and more. Gorm allows developers to easily create and edit graphical interfaces with many GUI elements easily and efficiently. Gorm is an application and thus is part of the application layer of GNUstep. It is used by developers to design interfaces visually. It interacts primarily with the GUI layer (libs-gui), but is itself, at the top of the architecture in the application layer.

Gorm includes gormcore, which is the central framework of the gorm application. It provides all the essential classes needed to interact with gorm files. It is the backbone of gorm, which enables graphical interface design. It is also used by gormtool, which allows developers to access gorm's features from the command-line. gorm also contains InterfaceBuilder, a framework that extends the functionality of gorm by allowing the developer to build custom palettes and inspectors. It acts as an extension of gorm to enhance developer experience.

4.2.2 Application Kit Layer (libs-gui)

Together libs-gui and libs-back are an implementation of the OpenStep Application Kit. The system works in such a way that there are little dependencies between GNUStep and the OS prioritizing cross-platform functionality. The libs-gui library provides a front end to the developer and makes display postscript (DPS) calls to the backend which then converts these calls to the underlying window system. Libs-gui and libs-back provide another layer in GNUSteps architecture where they interact with both the application layer and the foundation layer containing libs-base and make which it requires to run.

UI Elements

The GNUStep GUI library “libs-gui” is an object oriented component that contains many GUI interface classes. These classes include graphical objects such as buttons, text fields, popup lists, browser lists, windows, and more.

Window Management

Written completely in objective-C it is designed to cooperate with the libs-back backend framework to emulate the “the look and feel” of the display system without changing the application.

Drawing and Graphics

Libs-gui provides a comprehensive graphics system that allows developers to render high-quality graphics and UI components. The drawing engine uses Display PostScript (DPS) for rendering vector based graphics and supports gradients, transformations, image manipulation and colour management. This allows users to design without worrying about platform-specific rendering details.

Event Handling

Libs-gui framework supports a robust event-handling system to manage user interactions. The system supports mouse event tracking, keyboard event processing, gesture recognition, and responder chain management. By centralizing event handling, libs-gui ensures that all UI elements respond appropriately to user input, sustaining a good user experience.

4.2.3 Foundation Kit Layer (libs-base)

The base library of *GNUstep* (libs-base) is a “library of general-purpose, non-graphical Objective-C objects,” as described in the software’s official documentation. GNUstep base is the foundational component, for which other GNUStep components rely, providing data management, multithreading, and system interaction functionalities. It serves as the backbone of GNUstep applications, ensuring smooth communication between the software and the underlying operating system. The framework abstracts system-specific behaviors, which is critical in the cross platform functionality of GNU step, moreover it supports the evolvability of the software. Its modular structure allows for the addition of new features without breaking existing APIs, and system-specific code is isolated, simplifying the process of porting GNUstep Base to new platforms. Additionally, it includes a set of test cases to verify compliance with OpenStep and Foundation APIs.

Data Structures

Libs-base provides crucial data structure implementations such as Arrays, Dictionaries, Sets and Strings that are compatible with the OpenStep API, for the sake of portability.

Networking

Libs-base includes built-in socket programming, URL handling, and HTTP request functionalities, allowing applications to connect to remote servers and exchange data while maintaining cross-platform compatibility.

Threading

Multithreading is a core feature of libs-base, enabling applications to perform concurrent operations efficiently. It includes thread management APIs that support synchronization, locks, and concurrent task execution, ensuring that applications remain responsive even when executing complex background operations.

File Systems

libs-base provides file system abstraction to manage file I/O operations, directory traversal, and metadata retrieval across different operating systems. This ensures that file operations remain consistent regardless of the underlying system, simplifying cross-platform development.

Distributed Objects

GNUstep's libs-base includes a distributed object system, which allows inter-process communication and remote method invocation (RMI). This enables different components of an application, or even separate applications running on different machines, to communicate seamlessly. The system abstracts platform-specific communication mechanisms, ensuring a consistent API for developers.

4.2.4 Backend Layer (libs-back):

Libs-back corresponds to the GNUstep GUI back-end, and is a critical component of the GNUstep library responsible for managing platform-specific display system dependencies. It uses abstract low-level graphical operations, allowing GNUstep applications to run seamlessly across different display environments without the need for modification. The primary goal of libs-back is to provide an adaptable yet stable interface between GNUstep's front-end GUI components and various graphical systems such as X11 and Windows GDI. It separates platform-dependent rendering logic from application logic, allowing for portability, maintainability, and modularity. This design allows GNUstep applications to maintain a native "look and feel" while preserving cross-platform compatibility.

Libs-back is also configured to support future advancements in display technology. Due to the back-end's modular nature, additional back-ends can be implemented without affecting existing applications, and developers can optimize rendering strategies based on specific software or hardware environments.

Certain components of libs-back can be performance-critical, specifically in rendering operations and event handling. There is a distinct ongoing focus on optimizing graphical performance, especially in environments using the Display PostScript (DPS) emulation engine. These efforts include Cairo-based rendering, optimizing input handling, and refining resource management; all of which help to maintain responsiveness and efficiency.

Due to its integral role as an interface between the GUI library and display system, libs-back must be thoroughly tested across multiple platforms. By providing a flexible and efficient abstraction layer, libs-back allows GNUstep applications to remain adaptable to diverse environments, providing support for both legacy systems and modern graphical technologies.

4.2.5 Runtime Layer (libs-corebase)

Libs-corebase is a compatibility layer providing foundation-level functionality, such as CoreFoundation-like features. Libs-corebase enhances data handling and provides additional data structures and services that libs-back utilizes, such as font rendering, graphics, system event handling, etc. An extension of the base library, libs-corebase adds supplementary features that enhance cross-platform functionality. It includes additional data-handling capabilities and improved support for contemporary development practices.

The libs-corebase component of GNUstep serves as the foundation for low-level system operations, providing fundamental data structures and core utilities necessary for higher-level GNUstep frameworks. It is an open-source reimplementation of Apple's Core Foundation framework, aiming to ensure cross-platform compatibility while maintaining API consistency with macOS. This component enables developers to write Objective-C applications that function seamlessly across different operating systems, including Linux and Windows.

Libs-corebase plays a crucial role in GNUstep's architecture by acting as the foundation for essential system operations, implementing several key functionalities, and facilitating seamless integration with other components.

Data Structures

It provides essential container classes such as *CFArray*, *CFDictionary*, *CFSet*, and *CFData*, which facilitate structured data management. These structures are designed for performance and efficiency, enabling applications to store and manipulate large datasets while maintaining predictable memory allocation behavior.

String and Text Processing

The *CFString* class in libs-corebase allows for efficient Unicode string manipulation, including localization, encoding conversions, and case transformations. This component supports fundamental string manipulation.

Memory Management

The library employs manual reference counting, similar to Apple's Core Foundation, which requires explicit calls to retain and release objects, ensuring fine-grained control over memory usage.

Interprocess Communication (IPC)

It includes event-driven programming support through *CFRunLoop* and *CFNotificationCenter*, enabling asynchronous task execution, message passing, and event-driven application behavior. These mechanisms are crucial for applications that need to respond to user input or external events efficiently.

As a core dependency for higher-level frameworks, libs-corebase integrates closely with libs-base (gnustep-base), which implements the Foundation framework and serves as a bridge between libs-corebase and Objective-C applications. GNUstep's graphical framework, libs-gui (gnustep-gui), indirectly relies on libs-corebase through libs-base, utilizing its event-handling mechanisms for UI updates and user interactions. Similarly, libs-back (gnustep-back), responsible for rendering and low-level system interactions, depends on libs-corebase for efficient data handling and system event management. Additionally, Gorm, one of GNUstep's development tools, leverages libs-corebase to streamline project configurations, manage interface files, and handle user interactions effectively. By providing these foundational services, libs-corebase ensures the stability, portability, and functionality of the entire GNUstep ecosystem.

4.3 System Evolution

In 1996, the “GNUstep project” was initiated by the Free Software Foundation, with the goal of creating a free equivalent software to OpenStep. Early versions of the software (GNUstep 0.1 - 0.9) , released over the course of the next seven years, focused on implementing the foundations of the OpenStep API, but such versions lack stability and many OpenStep functionalities.

Version 1.0, the first stable release of GNUstep came in 2004, which implemented the OpenStep API in its entirety - notably replicating the Foundation and Appkit frameworks, which allowed users to build GUI applications in Objective C. GNUstep Version 2.0 arrived in 2007 and introduced the GNUstep 2.0 base library and increased compatibility with Apple’s Cocoa framework, which allowed developers to more easily port macOS applications to GNUstep. Version 2.0 also saw better integration with other operating systems and marked the release of the graphical interface builder tool, Gorm, which was meant to be an alternative to Apple’s interface builder.

Version 3.0 was released in 2011 and focused primarily on improvements in cross-platform functionality (Windows, Linux, etc.). Thereafter, Version 4.0 in 2013 marked improvements in the system’s functionality and efficiency through backend optimization of core components. Version 5.0, released in 2015, brought GNUstep into the modern world, made a series of updates to be most compatible with the newest operating systems. As well, Version 5.0 integrated GNUstep with other notable open source softwares like WebKit, which improved web-related functionality and performance.. The current iteration of GNUstep, Version 6.0 was released in 2016, with subsequent subreleases (6.1, etc.) focusing on maintaining and improving the software’s compatibility with modern platforms (Linux, MacOs, Windows) and work towards making GNUstep compatible with mobile systems.

4.4 Control and Data Flow

GNUstep’s control and data flow are structured in a way that allows for efficient interaction between the various layers, including the application, GUI, and foundation layers. Control flow is event-driven. User interactions, such as clicking buttons, entering text or dragging elements, trigger responses from the frameworks. These events propagate GNUstep’s libraries to allow applications to process the input, update the GUI and execute logic accordingly. The data flow involves exchanging structured information between components with objects, transferring messages and managing application data. The architecture ensures modularity and flexibility, allowing applications to be built by reusable components and communicate seamlessly.

The control flow of gorm is primarily user driven, developers manipulate UI components with a drag and drop interface. These interactions change the internal model, and the system dynamically updates the visual representation of the components. The developers can change aspects like size, position and function, while Gorm handles the backend configuration.The data flow to and from gorm involves storing, and changing UI configurations. When developers make changes to their interfaces, the UI attributes are stored in memory and converted into a .gorm file. This allows for automation and scripting, allowing for command-line integration.

Before the GUI is displayed at runtime the GUI library “libs-gui” needs to load the UI definition, read the .gorm file, and generate the UI elements by linking the UI objects to the objective-c code. Libs-gui hands off control to libs-base and libs-back for system level operations and UI rendering. Information is sent back to libs-gui to update the UI whenever needed. Once the UI is displayed the application can enter the event loop for user interaction. The data flows to and from libs-gui through the use of calls to the other libraries. In summary when a developer creates his GUI elements through gorm the GUI library will handle the creation of the elements through calling the other required libraries. If the created application is online and in an event loop then the GUI library will handle these events.

Libs-back serves as the back-end component for libs-gui, responsible for rendering graphics and managing low-level interactions with the display system. As libs-gui defines high-level interface elements, libs-back translates these into drawing commands that are system-specific based on the current environment. Libs-back runs a Display PostScript (DPS) emulator engine that interprets GNUstep’s drawing instructions on platforms that may not natively support DPS. At runtime, libs-back receives graphical and event-handling requests from libs-gui, and subsequently communicates with the native windowing system. At runtime, libs-back processes requests sent by libs-gui, fetches necessary system resources such as fonts and images and updates the display.

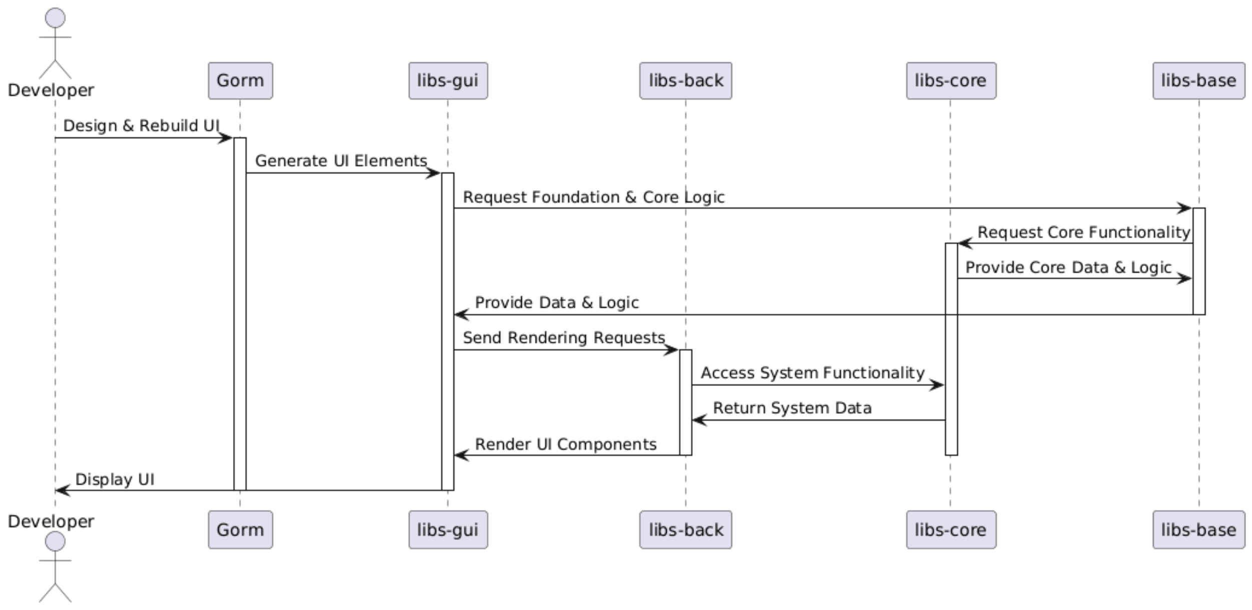
Libs-corebase provides the foundation for libs-gui and libs-back. It manages data structures, memory allocation and file handling when an application reads or writes a file, libs-gui sends a request to corebase, which processes the data and returns it to gui. It also handles any memory allocation or calculations before returning the results to the gui or back libraries.

5. Use Cases

5.1 Cross-platform Application Development

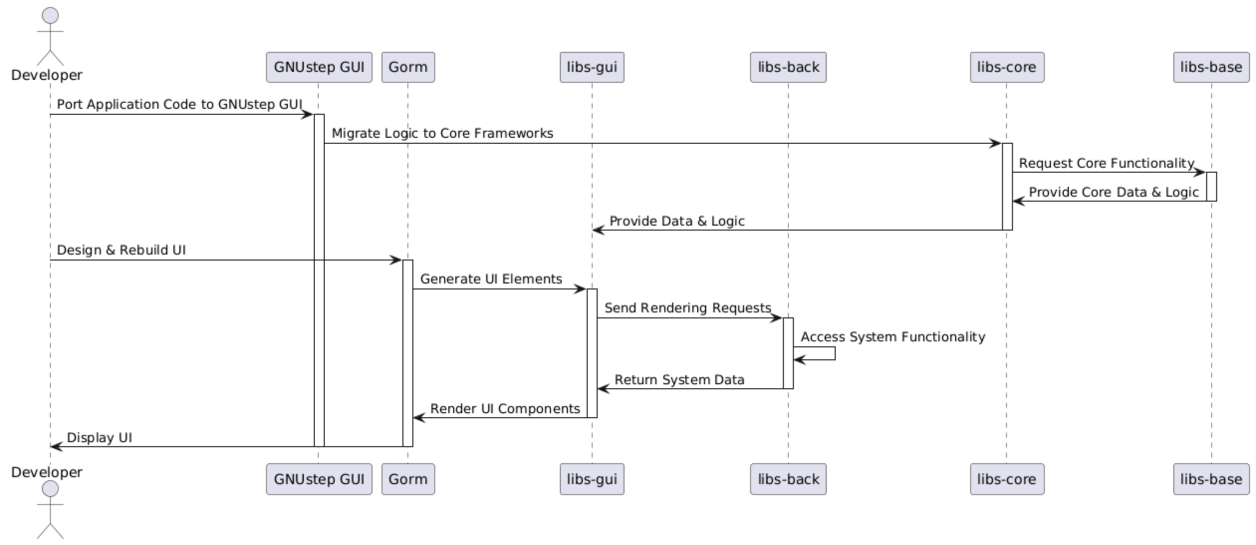
GNUstep enables developers to create Objective-C applications that run seamlessly across multiple platforms, including Linux, Windows, and macOS. By leveraging the fundamental system layer and graphical user interface layer, developers can design applications that maintain consistent functionality and user experience across different environments. This use case demonstrates how GNUstep's object-oriented architecture supports portability. The user interacts with gorm to create a UI using drag and drop functionality. These UI elements are made as objects through the interaction between gorm and the GUI library. The GUI library then calls upon the backend library to convert the UI into the appropriate systems style. When the selected UI elements have been created and converted appropriately they are rendered to the user based on the gorm design.

Additionally a user can add their own functionality and interaction through objective-c code. They can implement controllers, models, and connect outlets and actions from gorm to the main objective-c code.The code is then compiled into an executable with make which allows it to interact with GNUSteps libraries. The responsibilities are divided up like this: libs-base provides the system-level functionalities like files, networking, and data structures; libs-gui and libs-back handle the UI rendering and system integration; the windowing system and OS ensure the application is executed and displayed.



5.2 Modernizing Legacy MacOS Applications

Another use case of GNUstep is to modernize legacy macOS applications to allow them to run on windows and linux systems as well. If the application was built using Cocoa it can be ported using GNUstep. The development team simply needs to migrate the Objective-C code to GNUsteps libs-gui and libs-base frameworks. To ensure compatibility the user must modify macOS specific API's to the GNUstep equivalents. The UI is then rebuilt using gorm and after testing the ported application can be released on linux and windows. The various parts of GNUstep are used to port the application over while reusing as much of the old system as possible. Gorm is used to rebuild or redesign the old UI in a way that allows for the system to have little dependency on the OS itself. The libs-gui and libs-back extend this function, passing information and rendering the UI through the OS. libs-core and libs-base allow for the foundation of the application serving as an abstraction layer from the OS itself. The existing Objective-C code is repurposed to utilize these core frameworks of GNUstep to allow for more portability while maintaining the applications original functions.



6. External Interfaces

GNUstep operates on an external C runtime, specifically the GNU libobjc library, which is crucial for executing Objective-C programs. This runtime handles fundamental tasks such as message dispatching, memory allocation, and object lifecycle management. It plays a vital role in ensuring smooth cross-platform compatibility by managing low-level processes like function execution, exception processing, and efficient object handling.

GNUstep interacts with windowing systems and graphics libraries that are specific to the underlying operating system to provide the GUI functionalities. Some common systems thereof are X11 (on Unix-like systems), Win32 API (on Windows) and Quartz (on macOS). Graphics libraries like Cairo or OpenGl may also be used for rendering complex graphics in the GNUstep GUI environment.

It also integrates with external frameworks such as StepTalk (a scripting environment for GNUstep applications), Renaissance (a framework for declarative UI design), JiGS (a Java-based GNUstep interface), and GSWeb (a web application framework similar to Apple's WebObjects). These components work together by abstracting platform-specific details while maintaining high-level compatibility with the OpenStep and Cocoa APIs, allowing developers to build applications that seamlessly operate across different operating systems. Through these interactions, GNUstep ensures flexibility, modularity, and adaptability in modern software development.

7. Concurrency

GNUstep generally recommends developing single-threaded applications as they run faster and are easier to debug. That being said, GNUStep does have various multithreading capabilities. Threads can be created, synchronized, and locked with various classes from the Foundation Kit. NSThread allows for manually creating and managing new threads while NSOperationQueue allows for automatic task scheduling and executing tasks concurrently. Other classes like NSLock allow for better thread safety and synchronization by ensuring that two threads do not attempt to access the same critical resource at once.

8. Division of Responsibility

The division of responsibility among participating developers follows a two-role approach when working on the layered structure of GNUstep. GNUstep consists primarily of two different segments, the Foundation Kit and the Application Kit. The developers working on the Application Kit were focused on providing a set of classes oriented around creating and editing GUIs. This led to the creation of the GUI and back libraries which allow developers to create and use GUIs that do not rely on the base OS to function. The development team working on the Application Kit provided a layer of abstraction between the OS and the GUIs created by the users. The development of the Foundation Kit was focused on providing basic classes such as wrapper classes, and data structure classes.

Some developers were responsible for the integration of other GNU projects, creating new tools and functionalities. Since the structure of GNUsteps architecture leads to a division of specialized talent to different segments of the project, communication and documentation are extremely important. GNUstep has a well documented changelog as well as a wiki, a public github repository, and other resources to document the changes and updates to the project as well as planned events. With the core development of GNUstep complete the development plan has switched to a long term maintenance view where a team updates the libraries and frameworks of GNUstep as needed. To keep up with the evolving nature of Apple's proprietary software Cocoa, GNUstep is updated and extended as required.

9. Lessons Learned

Throughout the researching and writing stage of this report, we encountered challenges related to both gathering information and coordinating the division of work. Early on, finding detailed descriptions of the different aspects of GNUstep proved to be more difficult than previously thought. Assigning roles required continuous adjustments as we progressed with our writing and the report became clearer. As our vision became clearer though, we were able to refine our approach and divide the work more equitably. We maintained constant communication which allowed for these adjustments to be made in stride, though we could have defined our expectations earlier in the process for a smoother workflow. In hindsight, starting our planning phase sooner could have helped us to see any roadblocks sooner and we could have reduced the need for adjustments later in the process.

10. Conclusion

GNUstep's architecture embodies a carefully structured approach to cross-platform application development. By leveraging object-oriented programming principles and modular design, it provides a powerful alternative to proprietary frameworks. The core libraries, backend system, and GUI components work together to create an environment that fosters portability, maintainability, and innovation.

As open-source software, GNUstep continues to evolve, driven by contributions from a global community of developers. Its flexibility and extensibility make it a valuable tool for those seeking a reliable and efficient framework for building modern applications. Through its commitment to OpenStep principles and compatibility with contemporary technologies, GNUstep remains a robust and forward-looking platform for software development.

11. References

"GNUstep Base Documentation." *GNUstep*, Free Software Foundation, www.gnustep.org/resources/documentation/Developer/Base/Reference/.

"GNUstep System Overview." *GNUstep*, gnustep.made-it.com/SystemOverview/index.html.

"GNUstep." *Wikipedia*, Wikimedia Foundation, en.wikipedia.org/wiki/GNUstep.

"GNUstep." *GNUstep*, gnustep.github.io/.

"GNUstep Main Page." *GNUstep MediaWiki*, mediawiki.gnustep.org/index.php/Main_Page.

"GNUstep Developers Documentation." *GNUstep*, Free Software Foundation, www.gnustep.org/developers/documentation.html.

"GNUstep Developer Tools." *GNUstep*, Free Software Foundation, www.gnustep.org/experience/DeveloperTools.html.

"GNUstep ProjectCenter." *GNUstep*, Free Software Foundation, www.gnustep.org/experience/ProjectCenter.html.

"Gorm: A GUI Builder for GNUstep." *GNUstep*, Free Software Foundation, www.gnustep.org/resources/documentation/Gorm.pdf.

"Gorm Manual." *GNUstep MediaWiki*, mediawiki.gnustep.org/index.php/Gorm_Manual.

DeVries, Andrew. "GNUstep Manual." *Clemson University*, andrewd.ces.clemson.edu/courses/cpsc102/notes/GNUStep-manual.pdf.

"GNUstep Gorm." *GNUstep*, Free Software Foundation, www.gnustep.org/experience/Gorm.html.

12. Data Dictionary

API - A software interface that connects computer programs or computers, allowing different software components to exchange data and perform actions

GUI - a visual way for users to interact with a computer or website

13. Naming Conventions

GUI - Graphical User Interface
Libs - Libraries
DPS - Display PostScript
Obj-C - Objective-C
API - Application Programming Interface

OS - Operating System