CISC322 A1

Conceptual Architecture of GNUstep.

14, February, 2025

Quantum Loop

Mohamed Hirsi	22xlb@queensu.ca
Mirwais Morrady	22jl75@queensu.ca
Musdaf Hirsi	22pmw@queensu.ca
Daniel Bajenaru	21dsb12@queensu.ca
Mo Yafeai	21my32@queensu.ca
David Fabian	21dgf4@queensu.ca

Table of Contents

1 Abstract

GNUstep is a free and open-source application development framework built using the Objective-C programming language aimed at implementing the OpenStep API specification. Historically, OpenStep was an API designed by NeXT to simplify and streamline the development of platform independent GUIs and software applications[1]. GNUstep is modular by nature and can be installed on UNIX like systems as well as Microsoft Window. Its development is handled by volunteers and is overlooked by a team of core developers responsible for leading its development. With this background in mind, understanding the functionality of GNUstep becomes a matter of understanding what libraries are provided by the framework, and how developers use them to build applications.

2 Introduction

GNUstep is a constantly evolving cross-platform development environment for creating sophisticated and uniformly behaving graphical applications, as well as powerful command-line applications. Its inception began in 1994 following the public release of the OpenStep API specification. Ever since, GNUstep has consistently implemented and expanded upon every version of OpenStep. GNUstep is not the only implementation of the OpenStep specification. The creator of the specification, Steve Jobs' NeXT in collaboration with Sun Microsystem, had their implementation built into the OPENSTEP operating system. As history progresses, Steve Jobs returned to Apple which bought NeXT and developed a new version of the OpenStep specification calling it Cocoa. GNUstep thus strives to be compatible with Cocoa implementing as much of its updates as possible.

What separates GNUstep and Cocoa is their philosophy on public usage. GNUstep is a member of the GNU Project inheriting its core philosophy of being open-source and free for the general population. Cocoa on the other hand is proprietary to Apple's ecosystem of operating systems found on all their devices. As a result, GNUstep and all applications written using it can be installed and run on virtually every UNIX like operating system as well as Microsoft's Windows. Many GNUstep applications can be run on Mac OS X as well due to GNUstep's diligent trailing of Cocoa's updates. Naturally, Apple's applications can only be found within their ecosystem walling them from many computing users.

The backbone of GNUstep's functionality and performance is the Objective-C programming language. This is a minimal expansion of the well-known C programming language focusing to include the object-oriented programming paradigm with as few syntax changes as possible; as a result, there is little overhead to Objective-C when compared to C. The entirety of the GNUstep framework is built and using Objective-C and is fast and efficient due to the design of C.

The architecture of GNUstep follows the Object-Oriented architectural style. One of the key features of this style is its modularity and scalability. Objects are self-contained and independent of one another; all their data and methods are created and destroyed along with their own creation and destruction. Developers have the freedom to use as much or as little of the API in their development as it is partitioned into sub-libraries of related classes. The libraries include classes for graphical software called Applications, and non-graphical software called Tools.

3 Architecture

3.1 General Architecture

GNUstep implements the Object-Oriented architectural style. This is evident by the nature of GNUstep being an API. The various libraries found within contain classes and variables whose behaviors can be accessed only by the creation of objects. Each object is independent, meaning that they can be created and destroyed without affecting the existence of any other. These objects are the components of the architecture style, and they connect by enacting each other's methods.

There are two main functionalities for GNUstep, to develop Tools and to develop Applications. Tools are defined as non-graphical applications that are run using the command line[2]. They are usually helper applications or daemons, and do not have any resource files attached to them. Applications on the other hand are graphical applications which have graphical resources such as icons and images attached to them. Both functionalities leverage the makefile package, a tool for simplifying the process of creating makefiles which are crucial for speedy development with minimal hassle. Makefiles automate the process of building an application and allow for customization of the building process.

The primary use case for GNUstep comes from its GUI framework which gives developers a large set of classes to make sophisticated, uniform, and powerful graphical applications. This is all achieved by the libs-gui library along with supporting libraries such as libs-back. The classes in libs-gui handle objects such as "buttons, text fields, popup lists, browser lists, and windows.'[5] GNUstep's backend component is implemented in libs-back which provides functionality for handling system calls and communication with the windowing system.[6] Since GNUstep is designed to be platform independent, graphical applications maintain the look and feel of the user's system without the need for altering the code. This is because libs-back interacts with the user's windowing system directly, and libs-gui has platform independent classes. All that is needed to run a GNUstep application is to install GNUstep and the application which can be done on any UNIX like system and Microsoft's Windows. Although not guaranteed, almost all applications built using GNUstep are compatible with Apple's implementation of the OpenStep standard, Cocoa.

GNUstep provides a developer with many powerful libraries to create command-line applications as standalone projects, or as helpers for later app development. The libs-base and libs-corebase libraries as the backbone of all application development, but especially tool development. Both libraries work together in developing command-line applications. The corebase library targets lower-level aspects of a programming such as, "abstractions for common data types, facilitates internationalization with Unicode string storage, and offers a suite of utilities such as plug-in support, XML property lists, URL resource access, and preferences."[3] The base library on the other hand implements the non-graphical aspects of the OpenStep standard such as, "classes for strings, object collections, byte streams, typed coders, invocations, notifications, notification dispatchers, moments in time, network ports, remote object messaging support (distributed objects), and event loops."[4] The corebase library can be viewed as the skeleton of the GNUstep framework, and the base library as the muscles. Together, they give movement to any application built using GNUstep, and they are sufficient to have a complete and useful application without any graphical components.

3.2 Control and Data Flow

Control and Data flow in GNUstep (Wikipedia, 2025) are based on design principles from Object-Oriented Programming and the Objective-C language. They are handled in GNUstep by the Model-View-Controller design pattern, the target action paradigm, delegation, message passing, and the drag and drop concept.

3.2.1 Model-View-Controller (MVC)

Model-view-controller is a design pattern used for user interfaces that divides the program's logic into three elements: the model, the view, and the controller. The model component is the central part of the design. It directly manages the data, logic, and the rules. It keeps the data organized and consistent and ensures that it behaves properly according to the defined rules and logic. The view component represents visualizations of information (charts, diagrams, tables, etc.). It presents the model in a particular format. The controller takes input using that input to create commands for either the model or the view (Wikipedia, 2024a).

3.2.2 Target Action

The target action design pattern is used to divide programs into objects which tell each other which object to target and what message/action to send to that target when an event happens. The NSControl class in GNUstep includes key methods like `setTarget` and `setAction`, which provide the target action mechanism (Christley & Frith-Macdonald, 2025).

3.2.3 Delegation

Delegation is an object-oriented design pattern that allows one object to hand over a task/responsibility to another object. This helps with reusing code and flexibility. The NSApplication class provides delegation with methods like `setDelegate` (Christley et al., 2025).

3.2.4 Message Passing

Message passing is a technique where a program sends a message to a process, and that process then selects and runs the appropriate code to complete a task, rather than calling a program or function directly by name. This is done to achieve encapsulation, which is the idea that objects should be able to utilize services without knowing exactly how those services are implemented. This can reduce the amount of logic needed and can make systems more maintainable (Wikipedia, 2024b). The NSInvocation class in GNUstep allows for message passing by constructing messages, sending them to other objects, and handling the returned values (Frith-Macdonald & McCallum, 2024).

3.2.5 Drag and Drop

Drag and drop is a user interface feature where the user can select objects and drop them to various locations. This can be used for many different actions, like manipulating files (copying them into folders) or dropping files on applications (adding an image to a word document) (Wikipedia, 2024c). The NSPasteboard class in GNUstep can be used to implement drag and drop for transferring data (Frith-Macdonald, 2025).

3.3 Evolution

The evolution of GNUstep can be traced through several key stages, starting with NeXTSTEP. NeXTSTEP was an advanced operating system developed by NeXT, a company founded by Steve Jobs after leaving Apple. It combined a Mach-based kernel, Unix tools, and an Objective-C programming environment. It was the first system to introduce the AppKit framework, which later influenced Apple's macOS (Wikipedia, 2024). To make NeXTSTEP more widely available, NeXT and Sun Microsystems collaborated on OpenStep, a formal API specification that could run on multiple platforms, including Unix and Windows (MediaWiki GNUstep, 2024). OpenStep introduced a fully object-oriented development framework and GUI design principles, which later formed the foundation of macOS's Cocoa API (Wikipedia, 2024).

The GNU Project launched GNUstep as an open-source implementation of the OpenStep specification. Initially, GNUstep aimed to replicate the OpenStep API for Unix-based systems, but it later incorporated Cocoa-like features to enhance compatibility with macOS applications (GNUstep Official Documentation, 2024). Following Apple's acquisition of NeXT in 1996, the OpenStep API was integrated into macOS as Cocoa. Over time, Cocoa introduced new features such as Core Data, Grand Central Dispatch (GCD), and Swift support, which GNUstep attempted to incorporate (Wikipedia, 2024). While GNUstep has maintained strong compatibility with Objective-C, full compatibility with modern Apple frameworks remains a challenge (GNUstep Developer Tools, 2024).

GNUstep continues evolving, focusing on cross-platform development, modern Objective-C support, and GUI improvements. It remains an essential tool for developers who want to write portable Objective-C applications outside Apple's ecosystem (GNUstep GitHub, 2024).

3.4 Development

Being a large-scale open-source project, GNUstep relies on a well-organized division of responsibilities to keep development running smoothly. By distributing work among various groups, GNUstep ensures steady progress, maintains quality, and improves collaboration (GNUstep Developer Documentation, 2023). The way responsibilities are divided has a direct impact on efficiency, decision-making, and the project's overall stability.

3.4.1 Developer Roles and Responsibilities

Core developers are responsible for maintaining and improving the fundamental GNUstep libraries. Their primary role is to ensure that the framework remains stable, efficient, and compatible with OpenStep and Cocoa APIs. They work closely with other contributors, reviewing code and making major architectural decisions (GNUstep GitHub Repository, 2023). The following paragraphs highlight the main groups of developers and their responsibilities.

GUI and tool developers focus on improving the usability of GNUstep applications such as Gorm (the Interface Builder) and ProjectCenter (the development environment). Their goal is to make the framework more accessible and user-friendly for developers and users (GNUstep Wiki, 2023).

Platform compatibility engineers ensure that GNUstep runs smoothly across multiple operating systems, including Linux, BSD, and macOS. Their role is to integrate the framework with different environments, reducing fragmentation and increasing overall stability (GNUstep Contribution Guidelines, 2023).

Security and bug fixers play a vital role in maintaining the reliability of GNUstep. They focus on identifying and resolving vulnerabilities, ensuring that the framework remains secure. Their work requires close coordination with other teams to implement patches without introducing regressions (GNUstep Developer Documentation, 2023).

Documentation contributors ensure that GNUstep is well-documented, making it easier for new developers to understand and contribute to the project. They write and maintain guides, tutorials, and API references, bridging the knowledge gap between new and experienced developers (GNUstep Wiki, 2023).

3.4.2 Coordination and Collaboration

As mentioned above, collaboration is critical to the maintenance and further development of the GNUstep project. Mailing lists such as `bug-gnustep@gnu.org` and `gnustep-dev@gnu.org` are employed to discuss issues, propose changes, and seek advice (GNUstep Mailing Lists, 2023). Version control platforms like GitHub and GitLab host the codebases, enabling developers to submit patches and review each other's work (GNUstep GitHub Repository, 2023).

To maintain performance, all contributions go through code reviews by experienced developers. Changes must be documented in ChangeLogs, ensuring a clear history of modifications. Testing is also crucial; contributors are expected to verify that their code works before submitting it for review (GNUstep Contribution Guidelines, 2023).

3.4.3 Challenges in Role Distribution

Coordination Issues

While dividing work makes development more efficient, it also introduces challenges. Changes in one area—such as core libraries—can affect other components like the GUI or platform compatibility. Developers need to stay coordinated to prevent issues from spreading across the framework (GNUstep Mailing Lists, 2023).

Availability of Volunteers

GNUstep is primarily maintained by volunteers, which means contributor availability is unpredictable. Some areas may not receive enough attention, leading to delays in fixing bugs or implementing new features. Long-term maintenance of older components is also a challenge, as fewer developers may have expertise in legacy systems (GNUstep Contribution Guidelines, 2023).

Balancing Stability and Innovation

GNUstep aims to remain compatible with OpenStep and Cocoa APIs while also evolving with modern software needs. Developers must strike a balance between introducing new features and ensuring that existing applications continue to function without issues (GNUstep Developer Documentation, 2023).

Conclusion

To conclude the topic of coordination and collaboration, the structured division of responsibilities for GNUstep is essential for its continued success. By allowing developers to specialize in different areas, the project can progress efficiently while maintaining stability and security. However, effective collaboration, clear documentation, and an engaged community are crucial to overcoming challenges and keeping the project thriving.

3.5 Concurrency in GNUstep

GNUstep's concurrency model is primarily influenced by the Objective-C language and the Foundation framework, which provide mechanisms for parallel execution and asynchronous message handling. In this part of the report, we have tried to explore GNUstep's concurrency by examining key framework classes, asynchronous messaging, and Distributed Objects (DO).[24,26]

GNUstep provides several mechanisms for managing concurrency within applications. The primary aspects include:

- **Multithreading Support** – GNUstep offers support for multithreading, allowing developers to create applications that can perform multiple operations concurrently. This is facilitated through classes such as NSThread, which enables the creation and management of threads within an application.[22]
- **Operation Queues** – GNUstep base library's NSOperation and NSOperationQueue classes provide a higher-level abstraction for managing concurrent operations. They allow developers to define operations that can be executed concurrently, manage their dependencies, and control the maximum number of concurrent operations. This approach simplifies the management of complex concurrent tasks.[23]

- **Asynchronous Messaging** – Objective-C's messaging system allows for asynchronous message sending. We can see this in distributed object systems, where messages are sent between objects in different processes or across networks. The NSInvocation class plays a role here by encapsulating messages that can be forwarded or invoked at a later time. [24]
- **Distributed Objects (DO)** – GNUstep's Distributed Objects (DO) framework enables seamless communication between objects residing in different processes, whether on the same machine or across a network. This architecture abstracts the complexities of inter-process communication, allowing developers to design distributed applications as if all components were part of a single process.[25]

Examining Concurrency in GNUstep's Base Library

To understand concurrency in GNUstep, we examined key classes from the GNUstep Base Library. Below we take a short look at NSThread, NSOperation and NSOperationQueue as an example to understand of how GNUstep supports concurrency:

NSThread

NSThread provides low-level threading support, allowing developers to create and manage threads directly. Each process starts with a main thread, and additional threads can be spawned using NSThread. This approach offers fine-grained control over thread execution but requires careful synchronization to avoid issues such as race conditions and deadlocks. [22]

How NSThread Supports Concurrency

- Allows execution of multiple threads independently.
- Improves performance by handling multiple tasks in parallel.
- Requires explicit synchronization for shared resources.

NSOperation and NSOperationQueue

NSOperation and NSOperationQueue provide a higher-level abstraction for managing concurrency without dealing with low-level thread management. NSOperation represents a unit of work, while NSOperationQueue efficiently schedules and manages operations. [23,24]

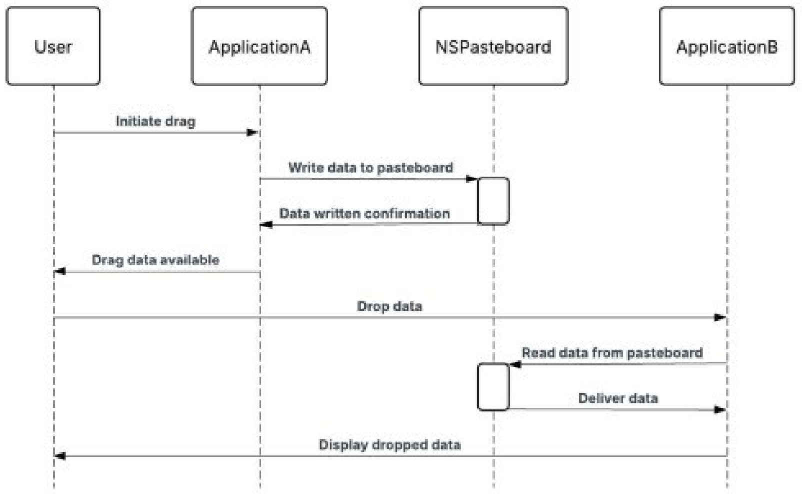
How NSOperation and NSOperationQueue Support Concurrency

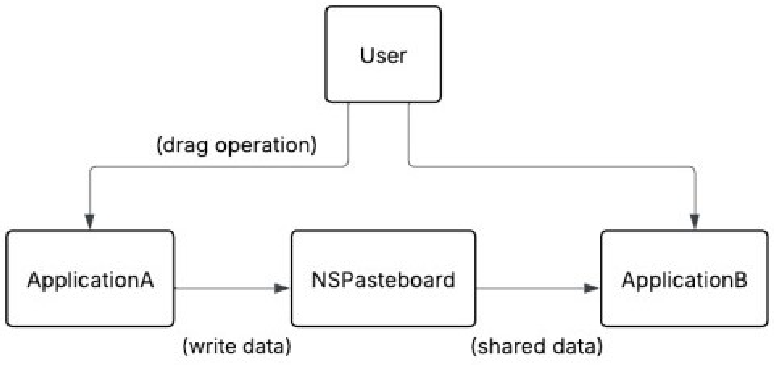
- NSOperation encapsulates a single task, allowing execution in parallel.

- NSOperationQueue handles the scheduling and execution of operations efficiently.
- Supports dependencies between operations, improving task management.
- Provides automatic thread management, reducing the complexity of concurrency handling.

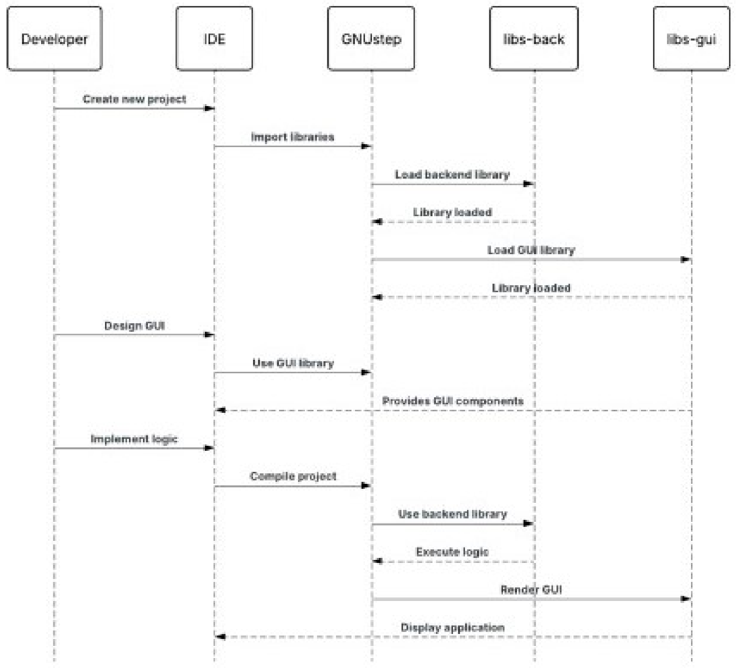
4 Diagrams

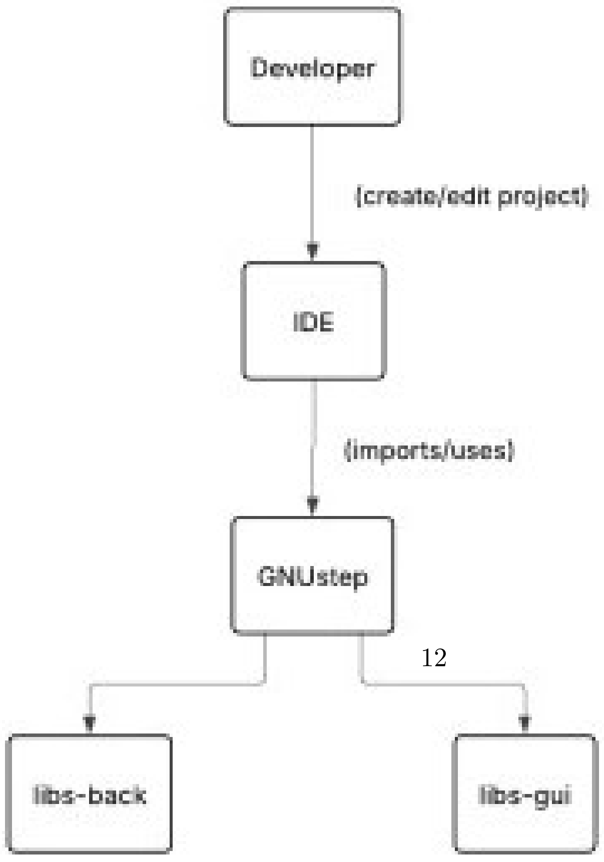
4.1 Use case: The user wants to drag and drop files between two applications.





4.2 Use case: The developer wants to create a GUI application using GNUstep's libraries libs-back and libs-gui.





5 External Interfaces

The main external interface that GNUstep uses is the GNUstep Database Library 2 (GDL2). It is a set of libraries that map Objective-C objects to rows of relational database management systems (RDBMS). It consists of the EOControl Library, which contains classes that handle the coordination of object graphs, the EOAcess library that implements the mechanisms of retrieving and storing data in RDBMS, and the EOInterface library which contains classes used to synchronize UI components with the state of objects (Viviani et al., 2025).

6 Use Cases

Use case 1: A user wants to drag and drop files between two applications. This use case utilizes a crucial part of GNUstep's control and dataflow (the concept of Drag and Drop). The use case activates the NSPasteboard class in GNUstep.

Use case 2: A developer wants to create a GUI application. One of the main functionalities of GNUstep is creating graphical applications, this activates many different components in GNUstep; two main libraries that would be used are lib-back and lib-gui.

7 Data Dictionary

Objective-C: An expansion of the C programming language which includes the object-oriented paradigm with as little changes to syntax as possible.

Graphical User Interface (GUI): A visual system that lets users interact with icons, buttons, menus and windows, instead of text-based commands.

Library: A collection of pre-written code containing functions and classes.

Framework: A collection of code libraries, headers, documentation and resources. They help developers by simplifying the development of software.

Application Programming Interface (API): A set of rules and protocols that allow different software systems to communicate and interact with each other.

Applications: Graphical applications that contain resources like icons or images.

Tools: Non-graphical or text-based applications that are run using the command line.

Command-line: A text-based interface that communicates with a computer's operating system. They are used to perform different tasks like running programs or manipulating files.

Open-source: Software whose code is publicly available and free for people to view, use, or modify.

Relational Database Management System (RDBMS): a program used to create, update and manage relational databases, which are systems used for organizing data into rows and columns to form tables.

8 Conclusions

Overall, GNUstep is a cross-platform framework for developing command-line and graphical applications using Objective-C. It includes libraries for handling low-level processes, graphical components, and communication to the system's windowing system. It is complete and standalone as well as free and open source.

9 Lessons Learned

One of the main lessons we learned was the importance of organizing team meetings before hand. Not all our schedules lined up perfectly so some of us couldn't make it to tutorial times which limited our ability of working on the report. To mitigate this issue, we decided to split the report into different parts that everyone could work on, and we utilized Microsoft Teams to keep all members updated. However, one thing we could have done differently is use the video/audio capabilities of Microsoft Teams to hold online meetings whenever possible. This would have led to better communication and organization throughout the group.

References

1. "OpenStep." *Wikipedia*, Wikimedia Foundation, `https://en.wikipedia.org/wiki/OpenStep`.
2. GNUstep Community. "Build Guide." *GNUstep Made-It*, `http://gnustep.made-it.com/BuildGuide/`.
3. Apple Inc. "Core Foundation Framework." *Developer Documentation*, `https://developer.apple.com/documentation/corefoundation`.
4. GNUstep Developers. "Core Libraries Repository." *GitHub*, 2023, `https://github.com/gnustep/libs-base`.
5. —. "GUI Libraries Repository." *GitHub*, 2023, `https://github.com/gnustep/libs-gui`.
6. —. "Backend Libraries Announcement." *GitHub*, 2024, `https://github.com/gnustep/libs-back/blob/master/ANNOUNCE`.
7. Christley, Scott, and Richard Frith-Macdonald. "NSControl Class Reference." *GNUstep Documentation*, 10 Feb. 2025, `http://www.gnustep.org/resources/documentation/Developer/Gui/Reference/NSControl.html`.
8. Christley, Scott, et al. "NSApplication Class Reference." *GNUstep Documentation*, 10 Feb. 2025, `http://www.gnustep.org/resources/documentation/Developer/Gui/Reference/NSApplication.html`.
9. "Drag and Drop." *Wikipedia*, Wikimedia Foundation, 17 Dec. 2024, `https://en.wikipedia.org/wiki/Drag_and_drop`.
10. Frith-Macdonald, Richard. "NSPasteboard Class Reference." *GNUstep Documentation*, 22 Jan. 2025, `http://www.gnustep.org/resources/documentation/Developer/Gui/Reference/NSPasteboard.html`.
11. Frith-Macdonald, Richard, and Kachites McCallum. "NSInvocation Class Reference." *GNUstep Documentation*, 18 Nov. 2024, `http://www.gnustep.org/resources/documentation/Developer/Base/Reference/NSInvocation.html`.
12. "GNUstep." *Wikipedia*, Wikimedia Foundation, 22 Jan. 2025, `https://en.wikipedia.org/wiki/GNUstep`.
13. GNUstep Community. "Developer Guidelines." *GNUstep Wiki*, 2023, `https://mediawiki.gnustep.org/index.php/Developer_Guides`.
14. —. "Project Contributions." *GNUstep Wiki*, 2023, `https://mediawiki.gnustep.org`.
15. "Message Passing." *Wikipedia*, Wikimedia Foundation, 13 Dec. 2024, `https://en.wikipedia.org/wiki/Message_passing`.
16. "Model-View-Controller." *Wikipedia*, Wikimedia Foundation, 8 Dec. 2024, `https://en.wikipedia.org/wiki/Model-view-controller`.

17. NeXT Software, Inc. *OpenStep Development Tools Manual.* NeXT, 1994, `https://cdn.preterhuman.net/texts/computing/nextstep-openstep/802-2110-OpenStep-Development-Tools.pdf`.
18. Viviani, Mirko, et al. "GNUstep Database Library 2 (GDL2)." *GNUstep Documentation*, 14 Feb. 2025, `http://www.gnustep.org/resources/documentation/Developer/GDL2/GDL2.html`.
19. "Contributor Guidelines." *GNUstep Official Site*, 2023, `https://www.gnustep.org/developers/contribute.html`.
20. "Developer Tools." *GNUstep Official Site*, 2024, `https://www.gnustep.org/experience/DeveloperTools.html`.
21. "Official Documentation." *GNUstep Official Site*, 2024, `https://www.gnustep.org/developers/documentation.html`.
22. " 'nsthread' Tag Wiki." *Stack Overflow*, `stackoverflow.com/tags/nsthread/info`. Accessed 10 Feb. 2025.
23. Ibanez, Andy. "Exploring the NSOPERATION Apis for Apple's Platforms." *Andy Ibanez - iOS Developer*, 21 Aug. 2019, `www.andyibanez.com/posts/exploring-the-nsoperation-apis/`. Accessed 10 Feb. 2025.
24. Dalrymple, David Chisnall. *The Objective-C Programming Language.* GNUstep, `www.gnustep.org/resources/documentation/ObjectivCBook.pdf`. Accessed 10 Feb. 2025.
25. Pero, Nicola. *Distributed Objects Programming: Introduction.* GNUstep, `www.gnustep.org/nicola/Tutorials/DistributedObjects/node1.html`. Accessed 10 Feb. 2025.
26. GNUstep. *GNUstep – Free Software Desktop and Development Environment.* `www.gnustep.org`. Accessed 10 Feb. 2025.