CISC 322/326
Assignment 1: GNUstep Conceptual Architecture
February 14, 2025

GNU16 Design Team
Group 16

Group Leader: Luc Robitaille - 21lar18@queensu.ca
Presenter 1: Enqi Liang - 21el32@queensu.ca
Presenter 2: Rodrigo Del Aguila Velarde - 20rdav@queensu.ca
Vlad Kukov - 21vk29@queensu.ca
Lucas Ferber - 19ljf1@queensu.ca
Robert He - 20rlth@queensu.ca

Table of Contents

1. Abstract

This report identifies and showcases the conceptual architecture of GNUstep. This report will only be analyzing specific libraries and applications connected to GNUstep. These libraries include the base, back, gui, and corebase libraries, and the application in question is Gorm. Also a note about this report is that many of these libraries interact with the make library. But we do not discuss this library in much detail as it was not listed in the description of major GNUstep components from the project description. The report starts off with a general overview of the conceptual architecture, where descriptions of GNUstep's high-level architecture and design style are discussed. Then this report goes down a layer and describes the conceptual architecture of the main subsystems within GNUstep. The relationships between these subsystems are also depicted during this section. Next the report goes in-depth into the evolution of the most important libraries within GNUstep, specifically the base library and the GUI library. Furthermore during this section, the evolution of the application Gorm is depicted as it is an important application used for GUI building, and something that this report is supposed to analyze. Next, the report looks into the data and control flow between the important libraries in GNUstep and the Gorm application. It was discovered when analyzing the data and control flow that the corebase library was created many years after GNUstep's initial release. This led us to realize that corebase should not be included in our conceptual architecture, as it was not part of GNUstep's conceptual architecture from the beginning. The following section determines if GNUstep has any concurrency within its framework, which we found to have concurrent processes. The succeeding section then looks into the responsibilities of contributors towards GNUstep, as GNUstep is open-source, it is important to understand the process behind contributions. Then this report discusses a couple of the important external interfaces that interact with the GNUstep framework. Next, this report presents two potential use cases of GNUstep, with sequence diagrams to visually show the reader how GNUstep handles the depicted use cases. Then this report lays out some of the important new terms used in a data dictionary and naming conventions section. Finally, this report discusses the lessons learned by all of us through this process, and our final conclusions on the conceptual architecture of GNUstep.

2. Introduction

The main purpose of this report is to look into and analyze the conceptual architecture behind the GNUstep framework. GNUstep is an open source object-oriented development environment originally derived from the OPENstep API and utilizing the NEXTstep framework that provides the user with a large range of utilities and libraries for building large, cross-platform, applications and tools. It is split into three components: Base, non-graphical classes corresponding to the NeXTstep Foundation API, GUI, consisting of graphical classes corresponding to the NeXTstep AppKit API, and Back, a modular framework for rendering instances of the GUI classes on multiple platforms. This enables developers to develop software that can run seamlessly on different UNIX-like platforms, including MACos. Through community effort GNUstep has continued to develop to meet modern standards.

3. System Architecture

3.1 Overview of Conceptual Architecture

GNUstep was conceptualised to provide an open source implementation of OpenStep, resulting in a conceptual architecture equivalent to its commercial counterpart. The Object Oriented Layer approach taken as it is ideal for implementing the cross platform approach GNUstep desired. Layers would allow much of its code to be reusable when transferred to a new windowing system, and objects are ideal for creating UI elements as they link better to real world objects. Cross compatibility was the responsibility of the Backend component. Which acted as a Layer separating the windowing system from the rest of the code. It could render and handle events to be compatible with a multitude of windowing systems, while allowing other components of GNUstep to be reused on any platform without having to be changed. The Application Kit would provide the graphical classes that would allow the user to create applications. It stored classes for events, fonts, colors, and many more UI elements. The Foundation Kit would provide non-graphical classes. Such as strings, collections, times, a random number generator, and other useful objects.

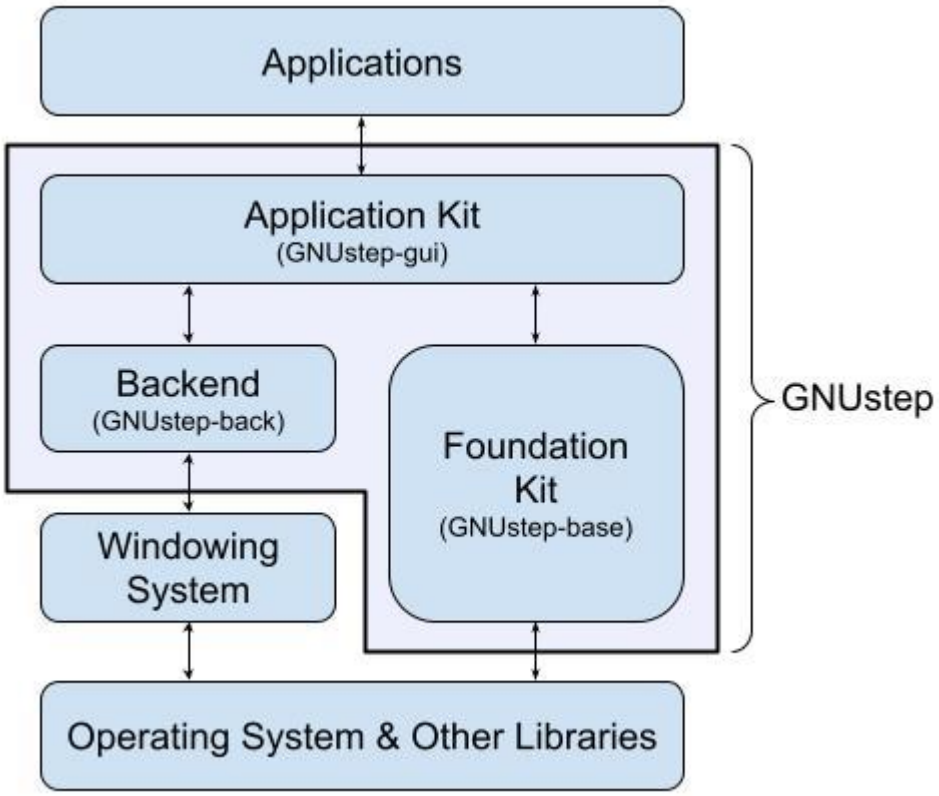


Figure: Diagram of the Data & Control Flows in GNUstep

3.2 Subsystems

3.2.1 Application Kit

The Application Kit defines Graphical Objective C classes. Its initial release included classes for events, fonts, colors, applications, menus, windows, views, controls such as

buttons, text fields, rich text, sliders, and popup buttons. All of these classes would be stored in a library called GNUstep-gui. This subsystem interfaces with applications outside of GNUstep. Such as with GORM, GNUstep's interface builder which was released a few years after GNUstep. A backend library is required for the Application Kit to function.

3.2.2 Foundation Kit

The Foundation Kit defines non-graphical Objective-C classes. Its initial release included classes for strings, collections, byte streams, typed coders, invocations, notifications, notification dispatchers, times, network ports, remote object messaging support (distributed objects), event loops, and random number generators. The Foundation Kit interacts with the GNU Objective-C runtime library. Which was being developed alongside GNUstep as part of the GNU project. The GNU Objective-C runtime library is not part of GNUstep, and belongs to another library.

The Foundation Kit is implemented in GNUstep as the GNUstep base. It is responsible for some of the fundamental actions for an application. It provides reusable code for functions including data representation and various forms of data storage. GNUstep is able to be compatible with both OpenStep and Cocoa by compiling different library header sections depending on the API. Additionally, execution is affected by the configuration files which determine where specific resources are installed and stored. The file paths can either be absolute or relative to the location of the configure file. The user is able to modify the file locations unless explicitly prevented by the system manager by removing the reference to the user configuration file in the main file.

GNUstep uses an object-oriented approach in its foundation. By default, an object cannot be copied and objects will only evaluate as identical if they reference the same address. Object containers including sets, arrays and dictionaries are immutable but have specific modified classes in which objects can be added or removed. Checking if a list is a subset of another, ordering a list and other operations are utilities which are possible with them. These containers cannot hold primitive variables so hashtables and maptables are used for non-objects. Alternatively, primitives can be stored as collections of bytes or can be converted to an object only representing the value. There also exist classes for structures including intervals and graph coordinates. GNUstep supports several string operations including parsing strings and extracting values with delimiters, adding attributes to strings and converting them to and from other data types.

GNUstep also supports file operations such as reading, writing, and modifying the location of a file. It is also possible to get all files below a specified directory. Data can be stored in a serialized property list but the classes of objects are not saved. Another form of data packing is archiving which saves an object instance and any references to it.

Communication between objects can be done synchronously or asynchronously depending on if a message queue is used. Interprocess communication is performed on a single device using a notification centre. A run loop is used to repetitively check for messages and pipes carry notifications from one task to another. Multithreading (multiple tasks at once with the same data) can be done with or without communication among processes and locks are used to ensure mutual exclusion on specific operations.

Communication between threads can be implemented using the same process as communication between distributed objects. GNUstep has classes for networking which keep host information, using URLs, serialization and messages sending as well as address storage and lookup.

Objective-C has variables which reference memory addresses which makes it unsuitable for distributed computing. GNUstep solves this issue by using a proxy variable which sends and receives messages from the server. In order to facilitate this, the server is made the root object of a connection instance and returns a variable after performing the requested command. The name of the connection is registered with the network and referenced in the variable as the proxy.

3.2.3 Backend

The Backend Subsystem is actually a series of libraries each designed for a specific windowing system. It uses a different library depending on the Windowing system in use. Its job is to handle rendering and events. The Application Kit is no use without it since graphical classes need something to render them. This subsystem separates the Application Kit from the Windowing System, allowing for the code from the Application Kit to be reused so long as the Backend supports the current windowing system.

4. External Interfaces

4.1 Objective-C Runtime & Compiler

Objective-C is an Object Oriented programming language and was the choice of language used to develop GNUstep. Code from GNUstep would need to be compiled with the Objective-C compiler into machine code before it could be understood by the operating system. Once compiled. The application can now be run on the computer. The role of communicating with the Objective-C library is done by GNUstep-base.

4.2 Gorm

Gorm is an interface modeler used to build and implement items from the GUI Library. The GNUstep GUI Library contains the classes for creating graphical applications, displayed through items which can be added and changed on the user interface. It was innovative due to its developer friendliness, as it was one of the first to use visual means to design user interfaces. The GUI is application centric meaning any new object instantiated is part of the same application system rather than another instance of the program. The applications are grouped in a “.app” directory which includes the code and resources for it. Interface files are used to display the UI and create connections to objects in order to make the program interactive.

4.3 Windowing System

The Windowing System is what a computer uses to render graphical applications on the screen of a computer. Since there are many windowing systems which work differently from each other. The Backend subsystem is able to check which Windowing System is in

use and convert objects from GNUstep-gui, to a graphical representation compatible with the windowing system. This graphical representation is then sent back up the layers to be displayed as part of the application.

5. System Evolution:

Development of GNUstep began in 1991 as a project to create a port of HippoDraw, a statistical data analysis package used for the analysis of particle physics data, that is usable on systems other than NeXt without having to completely recreate the application. Each of GNUstep's core libraries, libs-base, libs-GUI and Gorm, have each been developed independently with no clear timeline of release versions between components. As such, the evolution of each component will be covered separately.

Libs-base: Version 0.1.19 provided functionality of many new classes and objects and improved the functionality of preexisting classes. Several directories were reorganized and renamed to provide better organization of future classes. Several functions were added that provide the use of arrays and linked-lists. Version 0.5.5 implemented rewritten versions of many of NeXTstep's classes, improved so that they now ran at least as fast and in some cases faster than the NeXTstep implementations. This version also greatly improved the configuration and compilation of classes and made them easier to use for new users. Version 1.15.4 provided greatly improved compatibility for MacOS by creating many new methods and removing old methods that were not compatible with Mac. Version 1.23.0 adds full compatibility of Objective-C 2.0 and also adds automatic reference counts to manage object lifetimes which relieves the burden of having to manually manage references from developers. Version 1.31.0 is the current version of libs-base and continues to improve runtime and make minor improvements to various classes.

Libs-gui: Version 0.10 improves various classes and gui elements, such as a drag and drop feature and the ability to use various tab views. Many simple applications are now able to be used with GNUstep graphical interface with several NeXTstep apps and libraries being ported over. This version also includes various features such as, page up and page down and uses named system colours for easier configuration. This version also adds compatibility with several other applications including Gorm. Version 0.2 saw improvements to the optimization of the print function as well as the implementation of auto saving in documents. Further improvements were made with the compatibility of MacOS, with several new methods being added from Mac OS X 10.5. Version 0.32.0 is the current release and implements unique icons for various application folders as well as improved mouse tracking. This version also sees the implementation of Polish, German, Russian and Japanese language support.

Gorm: Version 0.1.0 implements the core functions of the application such as, adding the ability to drag elements to various windows, shift clicking to select multiple elements, a save and load function, various pop-up and pull down menu controls and the ability to change fonts. Version 0.5 sees the addition of sound and image support, multiple selection using mouse click and drag as well as greatly improved performance. Version 1.0.0 implements the ability to restore a deleted menu if it is deleted, an alert panel if a model fails to load or test, new widgets for font changing that make it easier for users to navigate, improvements to the cut and paste feature, various alignment tools which allows views to be

centered or moved between different view layers. Version 1.4.0 allows easier creation of matrix objects as well as the ability to add rows and columns.

6. Data and Control Flow

Data flow relates to the idea of representing different computations within a system and their interactions with a directed graph to show where data is coming and going between the parts of a system. Control flow relates to the idea of representing the order in which the systems that are interacting with each other are executed or evaluated. The assignment states to only look at 4 important libraries/components within the GNUstep system and the Gorm application. Due to the nature of this report focusing on the conceptual architecture of GNUstep, the library corebase is not included in this analysis of GNUstep, as this library was created in 2012. Therefore, this report will only discuss the data and control flow between these components and the Gorm application.

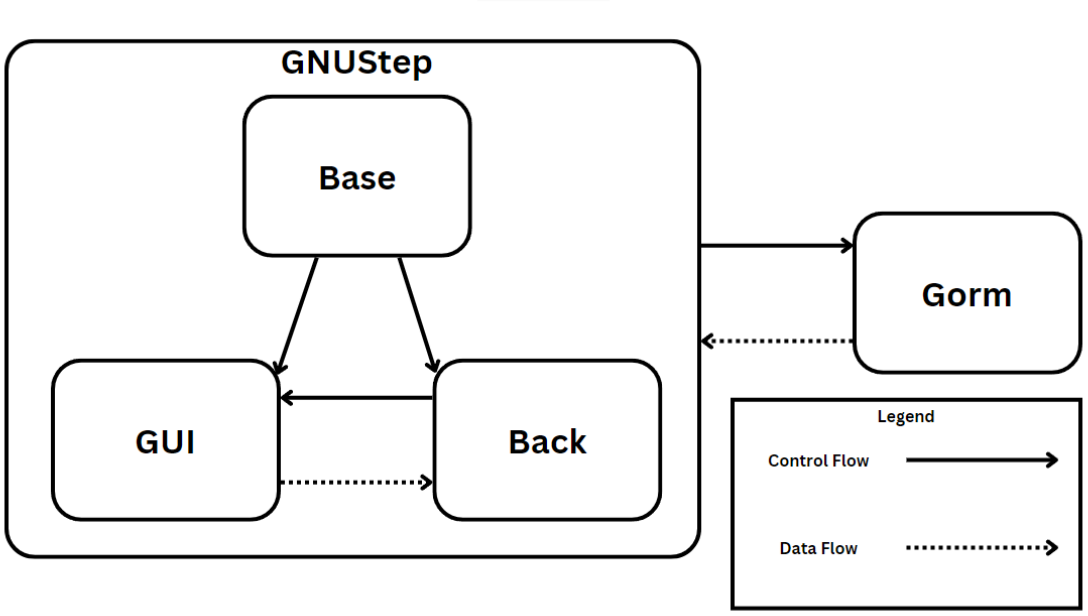


Figure: Data and Control Flow Diagram

The figure above showcases the data and control flow of the specified libraries and applications from GNUstep. The base library found within GNUstep is the foundation for the GNUstep API, which is why it has precedence over the other libraries within GNUstep. The back library has control flow towards the GUI library as the back library is the module in charge of everything related to the GUI library. This is why the GUI sends data to the back library, as it is in charge of the backend processes behind building the GUI. All of the libraries just mentioned are within the GNUstep API. Due to Gorm being a separate application that uses GNUstep, it is depicted outside of the GNUstep area. The Gorm application sends in data to GNUstep, usually input from the user, and in turn the GNUstep API has control over the Gorm application.

7. Concurrency

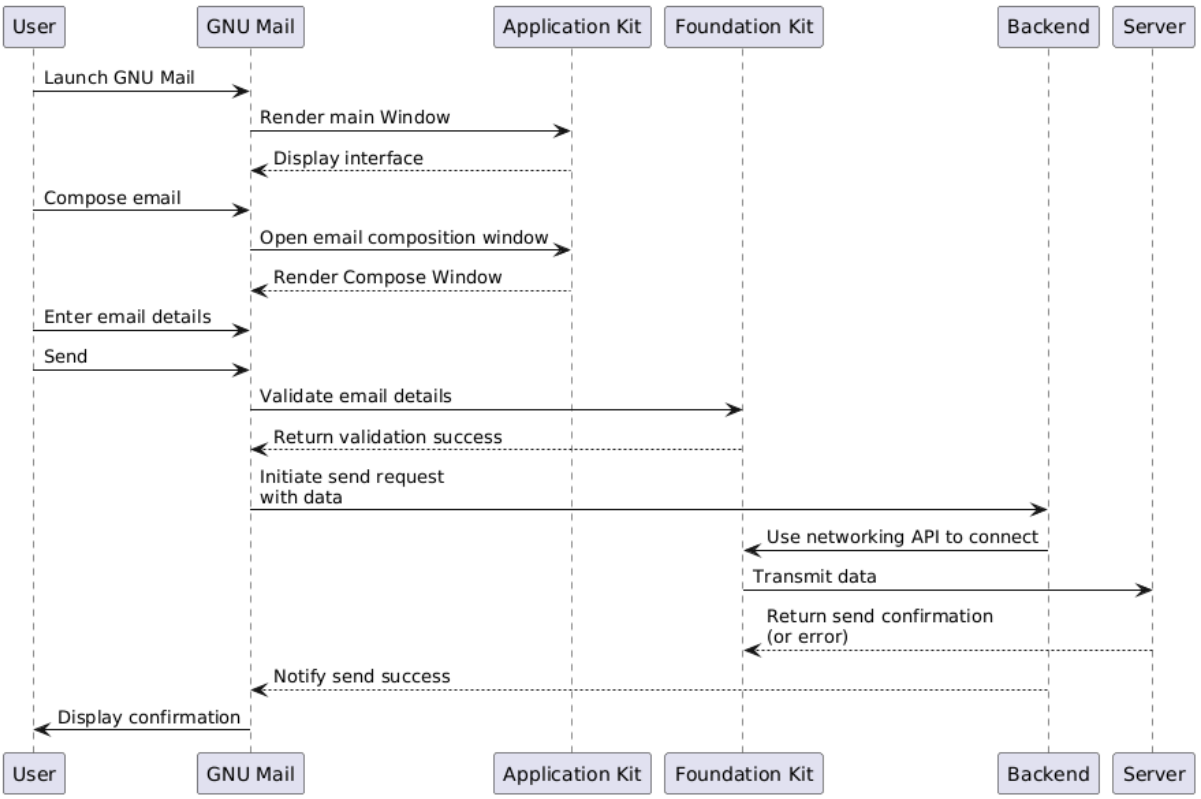
Due to the main purpose of GNUstep, it is an inherent quality that libraries need to be able run asynchronously for the user to be able to use GNUstep. This means that regardless if the process is single threaded or multi threaded, there is a form of concurrency present in GNUstep. For example, when the Gorm application is interacting with the GNUstep API, while Gorm is running, different libraries begin execution. This is a form of software concurrency, as multiple processes are operating at the same time. Also GNUstep itself is able to be used for creating multi-threaded programs, which is another example showing that GNUstep has concurrent properties.

8. Responsibilities Among Participants

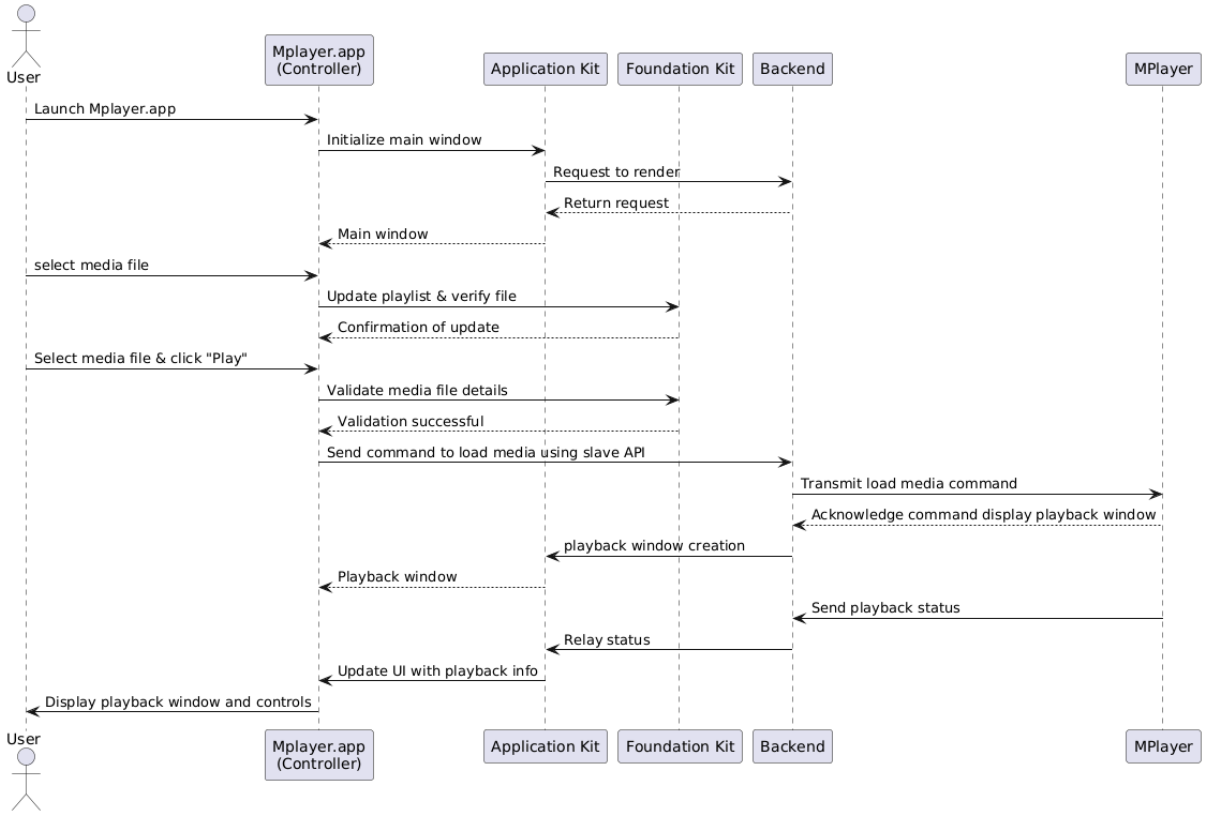
As an open source GNU project GNUstep is developed by a series of volunteers that simply choose a section of the project they wish to work on and then submit their changes. In order to maintain order within the project each non-trivial change to the project must be vetted by the project leads in order to ensure compatibility and to eliminate duplicate code. Each entry into the project must be properly documented with a list of changes. The changelog must include the date and the author's name. All documentation of code must be checked so that it is not too similar to the OpenStep or Cocoa documentation to avoid copyright infringement. All code must be written using Objective-C except a few exceptions where Objective-C 2.0 must be used. Developers must not use the Objective-C dot(‘.’) operator as it is undefined in some compilers. All memory management, variable declarations and naming conventions must comply with the GNU coding standards.

9. Use Cases

1. The first case involves using GNUmail to send an email. Where the user as the stakeholder tries to send, and the GNU mail must handle the input utilizing a graphical interface, verifying data and interacting with the GNUstep framework which provides cross-platform support, using application kit, foundation kit and backend. The user first opens GNU mail, which uses GNUstep’s application kit to render, then the user will open a new Email writing interface, also utilizing the application kit, then after the user has done typing, the user clicks the send button and the GNU mail will verify and validate the input data using the foundation kit. If the previous process passes, the GNU mail will send a send request. After the last step, GNU mail’s backend will establish a connection to the server and transfer the content in the email, receiving a confirmation message from the server, and then the message is shown to the user. If the process fails, it will notify the user.



2. The second use case is using the media player MPlayer.app in GNUstep to display media. The user first launches the Mplayer, which utilizes GNUstep's application kit, which utilizes the backend to render, and displays the user interface. The user adds a media, the controller will use the Foundation kit to update the playlist and verify the file. The user selects the media file and press play, then the Mplayer.app will send a command to Mplayer using its slave API to load the media file. Mplayer will provide a play window, displaying information to the user, and allowing the user to adjust settings.



10. Data Dictionary

Bundle - A group of resources which contain all files required for program execution such as code or images. The three variations are applications, frameworks and loadable bundles
Loadable bundle - A type of plug-in which can be loaded to extend an application. The two variations are plug-in and palette
Plug-in - A bundle which can be loaded to add functionality
Palette - A bundle which can be loaded to add UI objects to GORM
Application - A program with resources which are all contained in a single directory
Tool - A program with resources within the PATH environment variable making it executable from the command line, usually non-graphical
Framework - Shared dynamic library with resources such as documentation to be used by multiple applications

11. Naming Conventions

GS/GC - Prefix to classes present in GNUstep but not OpenStep
GSXML - Prefix for classes which read XML files for XPath expressions
GSHtml - Prefix for classes which read HTML documents
GSMime - Prefix for classes which operate with MIME messages or HTML POST documents
GSLazy - Prefix for classes which only lock data in a multithreaded application
GORM - Graphical Object Relationship Modeller

12. Lessons Learned

We have learned several lessons in the process of exploring the conceptual architecture of GNUstep. First of all we have discovered how difficult it can be to find formal documentation and description of a software application and its architecture. The challenges faced in finding resources to answer these questions taught us that discovering the structure of an application may require strong researching skills. It may explain why some open source softwares are abandoned, as software developers do not want to spend more time learning the system than actually contributing. Throughout the search we found many missing pages and some incomplete files which displays the lack of support. The lack of a cohesive specification made us appreciate all of the programs we have previously come across with well-documented architectures. It also warned and prepared us for future endeavors where we may find occupations which expect us to work on code with meaning that is difficult to decipher.

We have also learned several things about software architectures and app development. We discovered how a software for developing applications might work and what the processes involved are. It is valuable to learn about all of the different parts of a program that you might need to consider that we might not have previously considered. For example, the difference in communication between threads, objects within the same device and objects on another device was surprising. Previous knowledge of these topics would have definitely helped understanding.

Another lesson we learned was the importance of communication in large group projects. Although we can often get away with segregating certain sections of group assignments with smaller scopes, there are challenges working independently on complex deliverables with less obvious expectations. Regular discussions allow us to share information which may be tough to find or understand and ensure a clear and consistent report. This also has value for the future since it is likely that we will be working with teams on large scale projects in our future jobs.

13. Conclusions

The key findings from our research are that GNUstep's conceptual architecture is broken up into three main subsystems. The Application Kit (libs-gui), Foundation Kit (libs-base), and Backend library (libs-back). These libraries interact with each other in order for GNUstep to operate. Much of the conceptual architecture for GNUstep was based on the NeXTSTEP framework. The GNUstep framework interacts with many different external interfaces, with one of the most important being the Gorm application meant for creating visual GUI's. It took everyone a lot of digging for all of us to find all this information due to the age of GNUstep, so one proposal of ours would be for GNUstep to have more up to date resources describing the architecture of the system.

14. References

GNUstep History, gnustep.made-it.com/Guides/History.html.

“GNUstep Base Release Notes.” *GNUstep*, www.gnustep.org/resources/documentation/Developer/Base/ReleaseNotes/ReleaseNotes.html.

Gnustep. “Gnustep Libs-Base News.Texi.” *GitHub*, github.com/gnustep/libs-base/blob/master/Documentation/news.texi.

Fedor, Adam. “GNUstep Gui Release Notes.” *GNUstep*, www.gnustep.org/resources/documentation/Developer/Gui/ReleaseNotes/ReleaseNotes.html.

“GNUstep Apps-Gorm News.” *GitHub*, github.com/gnustep/apps-gorm/blob/master/Documentation/NEWS.

“Introduction (Coding Standards for Gnustep Libraries).” *GitHub*, gnustep.github.io/About/Contributing/gs-standards/Introduction.html.

“Dataflow.” *Wikipedia*, Wikimedia Foundation, 25 June 2024, en.wikipedia.org/wiki/Dataflow.

“Control Flow.” *Wikipedia*, Wikimedia Foundation, 6 Jan. 2025, en.wikipedia.org/wiki/Control_flow.

Botto, Francis, et al. *Objective-C Language and Gnustep Base Library ...*, andrewd.ces.clemson.edu/courses/cpsc102/notes/GNUStep-manual.pdf.

“Main Page.” *GNUstepWiki*, mediawiki.gnustep.org/index.php/Main_Page.

Armstrong, Christopher. *Using the Gnustep Appkit*, www.ict.griffith.edu.au/teaching/2501ICT/archive/resources/documentation/Developer/Gui/ProgrammingManual/AppKit.pdf.