https://youtu.be/0-bYd7HXX6M

Big Hero 6 Team

Team Members and Roles

Team Big Hero 6 - Group 30

- Terence Jiang - Group Leader, Use Cases
- Kevin Chen - System Evolution
- Alexander Zhao - Presenter, Implication for Division, Abstract
- Carter Gillam - Presenter, Concurrency, Glossary
- Kavin Arasu - Introduction & Overview
- Daniel Gao - Control & Data Flow

Conceptual Architecture of GNUstep program

February 13, 2025

Introduction to GNUstep

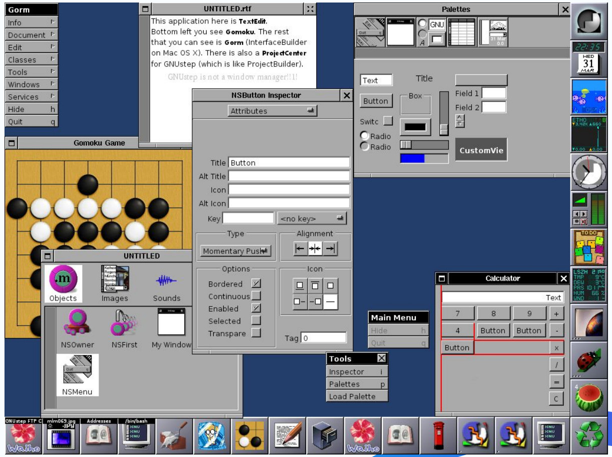
Goal of the Report: Analyze and define the conceptual architecture of GNUstep, an open-source implementation of the OpenStep framework for application development.

- GNUstep follows a modular, object-oriented architecture, with key components such as libs-back, libs-base, libs-corebase, libs-gui, and Gorm.

Component Functions:

- *libs-base*: Handles object management, networking, and file I/O.
- *libs-gui*: Implements Cocoa API functions for cross-platform GUI development.

GNUstep's Evolution: Maintains compatibility with NeXTSTEP and macOS Cocoa while integrating modern features, emphasizing modularity, reusability, and openness.

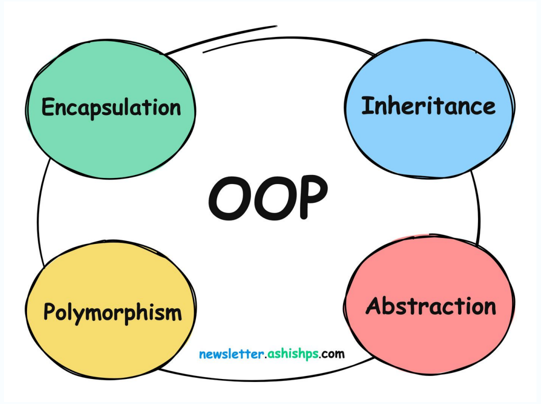


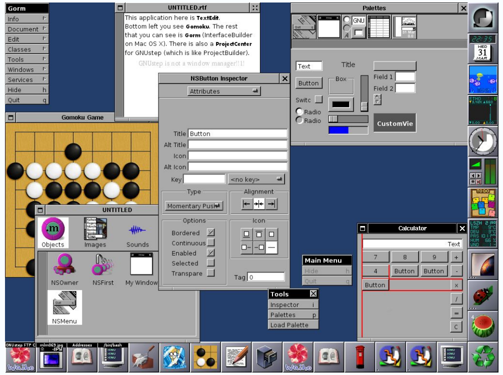
UNTITLED.rtf

This application here is **TextEdit**.
Bottom left you see **Gomoku**. The rest that you can see is **Gorm** (InterfaceBuilder on Mac OS X). There is also a **ProjectCenter** for GNUstep (which is like ProjectBuilder).

GNUstep is not a window manager!!1!

Architecture Style




This application here is **TextEdit**. Bottom left you see **Gomoku**. The rest that you can see is **Gorm** (InterfaceBuilder on Mac OS X). There is also a **ProjectCenter** for GNUstep (which is like ProjectBuilder).

GNUstep is not a window manager!!1!

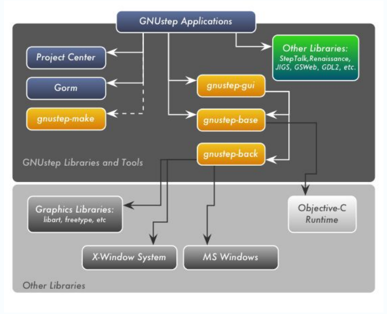
Framework Architecture

- Object-oriented design with modular components
- Modular components are very easy to maintain

Design Principles

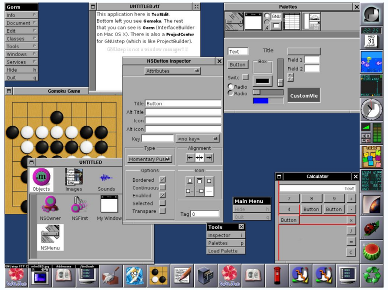
- Encapsulation and inheritance for code reuse
- Loose coupling between framework components
- Well defined interfaces for interaction

Key Components



Core Libraries

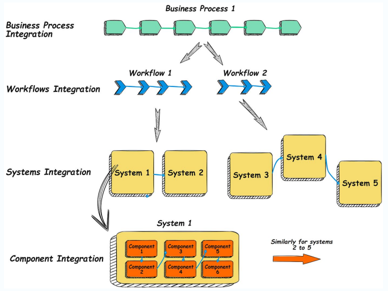
- **libs-base -** Core Objective-C classes and system functions.
- **libs-corebase -** Adds macOS-compatible foundation APIs.
- **libs-gui -** Provides AppKit for creating GUIs.



This application here is TextEdit. Bottom left you see Gomoku. The rest that you can see is Gorm (InterfaceBuilder on Mac OS X). There is also a ProjectCenter for GNUstep (which is like ProjectBuilder).

Development Tools

- **Gorm** - Visual interface builder for UI design.
- **ProjectCenter** - IDE for project management.
- Streamlines workflows for application development.



Additional Components

- **GNUstep Make** - Automates builds and linking.
- **libs-back** - Renders graphics for multiple platforms.
- **Workspace** - Desktop tools for file management.

System Evolution

Origins of GNUstep: Began as an open-source reimplementation of NeXTSTEP's OpenStep framework, aiming for API compatibility with early macOS versions and providing essential Objective-C libraries and a GUI toolkit.

Key Evolution Milestones: Introduced Gorm and Project Center, added Automatic Reference Counting (ARC), blocks, and enhanced cross-platform compatibility and GUI capabilities.

Major Version Updates:

- GNUstep 1.0: Basic GUI toolkit for Unix-based systems with limited functionality.
- GNUstep 1.2: Improved memory management, AppKit enhancements, but still lacked cross-platform compatibility.
- GNUstep 1.4: Added Windows/macOS support, Gorm for GUI building, and NSTableView/NSTextView for complex layouts, though missing advanced macOS APIs.
- GNUstep 1.8: Introduced Core Graphics-style and Quartz-style rendering, improving graphics and animations but lacking multi-touch and high-resolution display support.

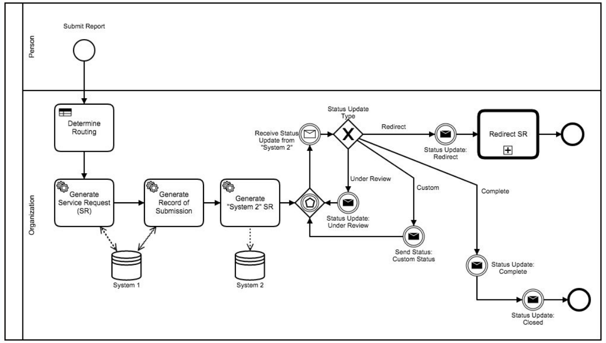
Control Flow

Application Kit (GNUstep GUI):

- Manages the main event loop, handling user input and sending events to corresponding handlers.
- Follows the Model-View-Controller (MVC) pattern:
    - Model: Stores and manages application data.
    - View: Displays the user interface elements.
    - Controller: Manages the logic, moderates data flow, updates the Model, and ensures the View updates correctly.

Development Tools:

- Gorm: A UI design tool enabling drag-and-drop element editing.
- Project Center: An IDE for writing and compiling source code.
- Both tools simplify application creation and management.



Data Flow Architecture

Foundation Kit (GNUstep Base):

- Handles data management, exception handling, and file system operations.
- Provides fundamental Objective-C classes for application logic.

Application Kit (GNUstep GUI):

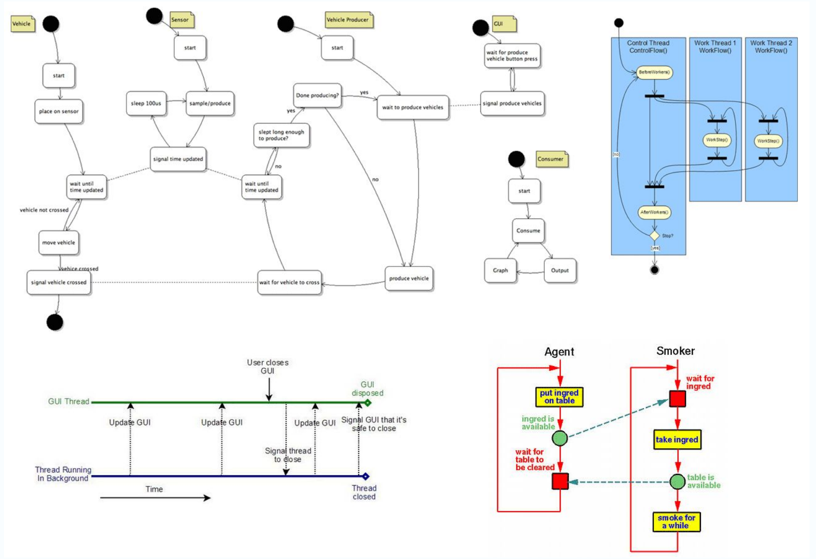
- Handles event-driven interactions, updating data based on user input.
- Ensures the Controller updates the Model and modifies the View accordingly.

Cross-Platform Support:

- GNUstep components work together to enable cross-platform applications with consistent data handling and UI behavior across different operating systems.

Concurrency Model

- Concurrency in GNUstep improves responsiveness, scalability, and reliability, making it essential for GUI applications and server-side operations that require real-time performance and fault tolerance.
- NSThread enables independent thread management, isolating resource-intensive tasks like file loading and calculations, while NSOperation and NSOperationQueue simplify task scheduling and allow background execution without freezing the interface.
- Fault isolation is a key advantage, as tasks run independently, preventing failures from affecting the main thread. Synchronization tools like synchronized blocks and locks prevent race conditions and ensure thread-safe access to shared resources.
- Challenges include race conditions, deadlocks, and performance overhead. Developers must carefully manage locking strategies to prevent deadlocks, while GNUstep minimizes overhead through lightweight thread management.



Developer Division & Organization

- GNUstep follows an object-oriented architecture, making it modularized, where components interact through well-defined interfaces. This allows independent development, maintenance, and replacement of modules without significantly affecting the program's structure.

- Development responsibilities are distributed based on expertise and module goals. Some developers work on lower-level Foundation classes like NSString, while others focus on graphical elements or tool development, such as enhancing Gorm for a better development experience.

- Inheritance and modular expansion enable new functionality by extending existing classes. Core libraries like libs-base, libs-corebase, libs-gui, libs-back, and gorm form the foundation, requiring developers to ensure compatibility and adhere to Cocoa API behavior rules.

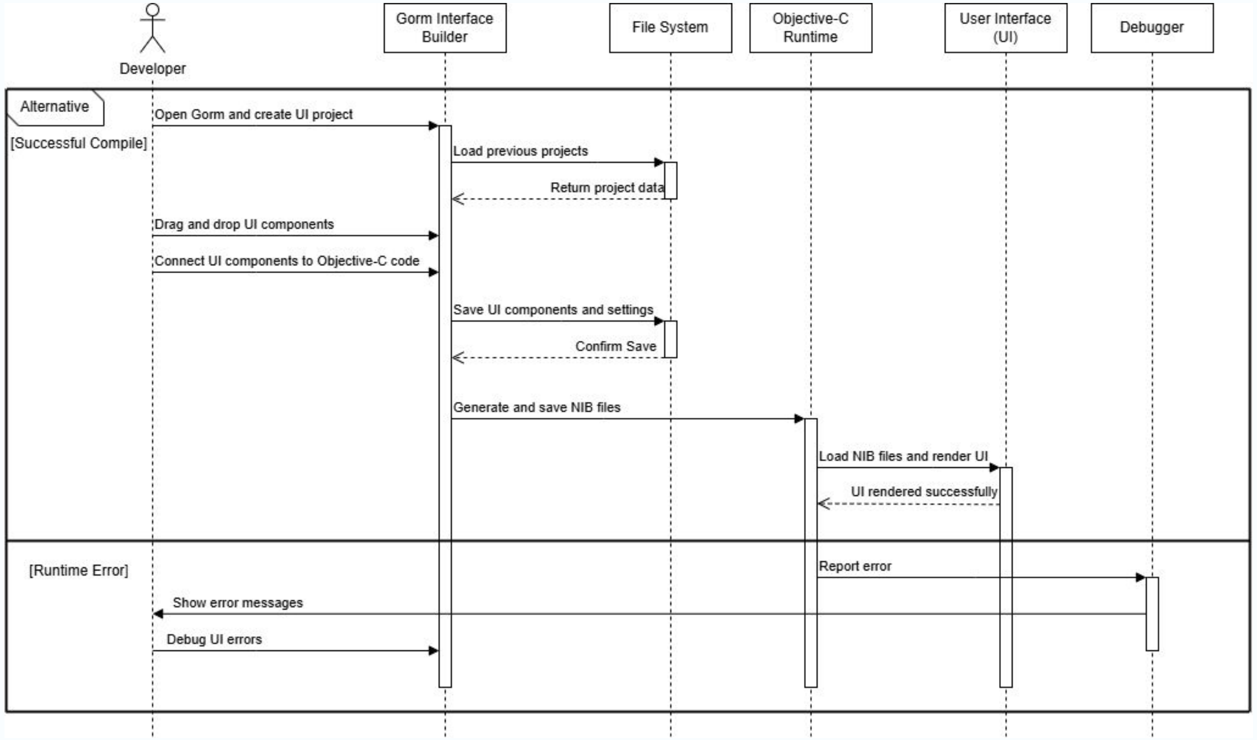
Developer Division & Organization

- Polymorphism within GNUstep demands careful maintenance, ensuring different classes interact seamlessly while preserving core functionalities. Documentation is crucial for helping contributors understand modules and their usage.

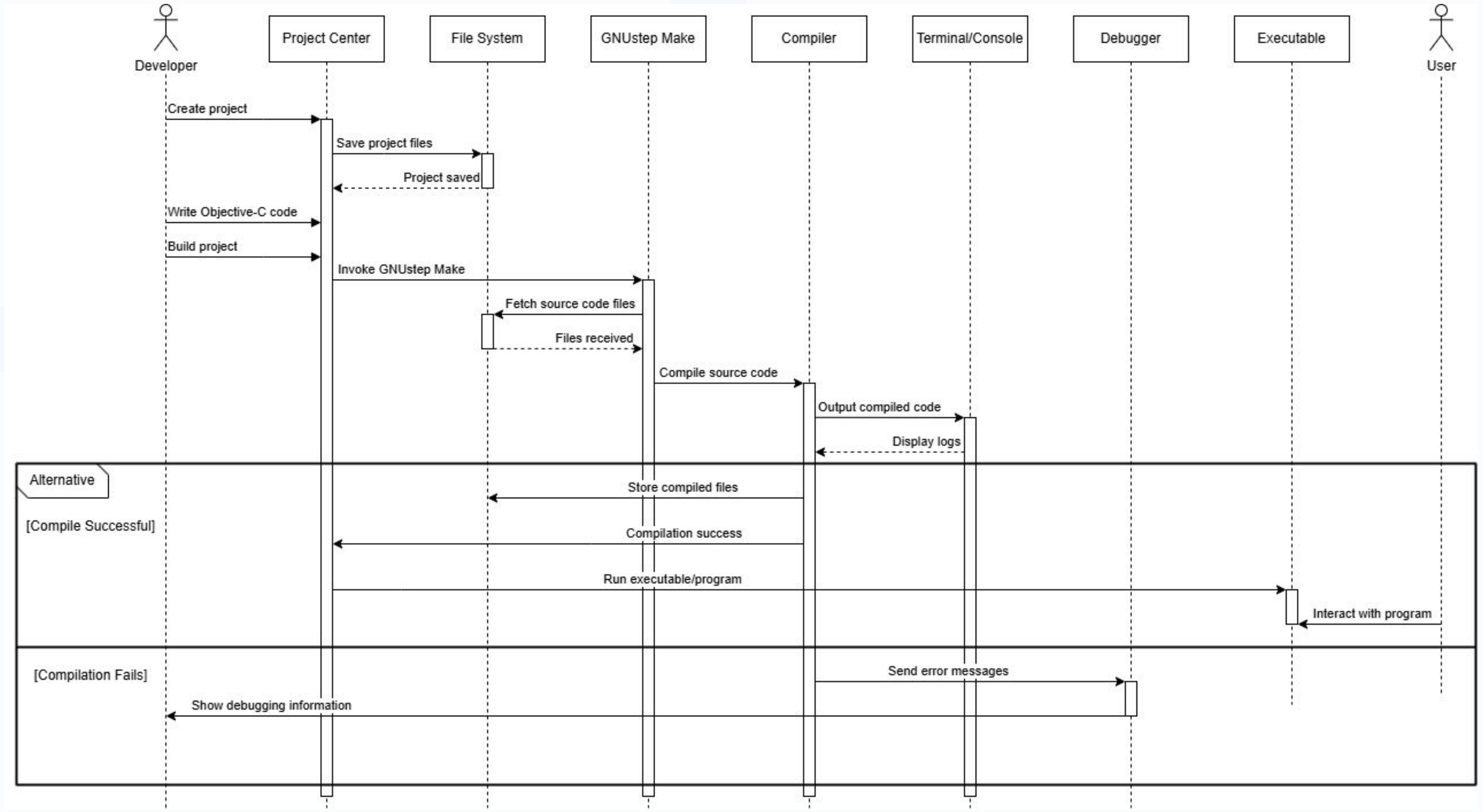
- Thread safety and concurrency must be considered, as some applications require multi-threading. Developers need to handle race conditions, resource sharing, and synchronization mechanisms to maintain stability.

- The division of responsibilities in GNUstep ensures modular development, consistency, and maintainability. Developers can specialize without deep system knowledge, Objective-C ensures consistency, and encapsulated libraries allow bug fixes and extensions without disrupting other components.

Use Case#1: Creating An App with Gorm



Use Case #2: Compiling and Running a Project in ProjectCenter



Architectural Insights

Key Architectural Strengths

- Modular design enables flexible component integration
- Strong compatibility with OpenStep standards
- Efficient memory management system
- Cross-platform portability and adaptability

Design Trade-offs

- Performance overhead in some GUI operations
- Limited modern UI framework features
- Resource intensive for small applications

Conclusion

GNUstep Architecture: Key Takeaways and Future

GNUstep's object-oriented framework provides a robust foundation for cross-platform development through its MVC architecture and comprehensive component system. While it excels in compatibility and modularity, developers should consider memory management and platform-specific optimizations. Future developments focus on enhanced mobile support, modern UI capabilities, and improved cross-platform performance.