Architectural Enhancement of GNUStep

GAMERS NEVER GIVE UP

Jawad Ahmed (Leader), Meagan Mann (Presenter), Ethan Nguyen (Presenter), Zoya Zarei-Joorshari, Kim Hyun Bin, Ripley Visentin

https://youtu.be/2uuv8csvPfE

Introduction

GNUStep is an open-source, cross-platform GUI development framework written in Objective-C, designed for compatibility with legacy and modern Apple systems.

We propose an architectural enhancement:
Gorm Cloud – a cloud-based extension to the Gorm subsystem.

This feature enables:

- **Remote GUI development**
- **Cross-team collaboration**
- **Improved scalability and ease of use**

Gorm Cloud: Enhancing GNUStep

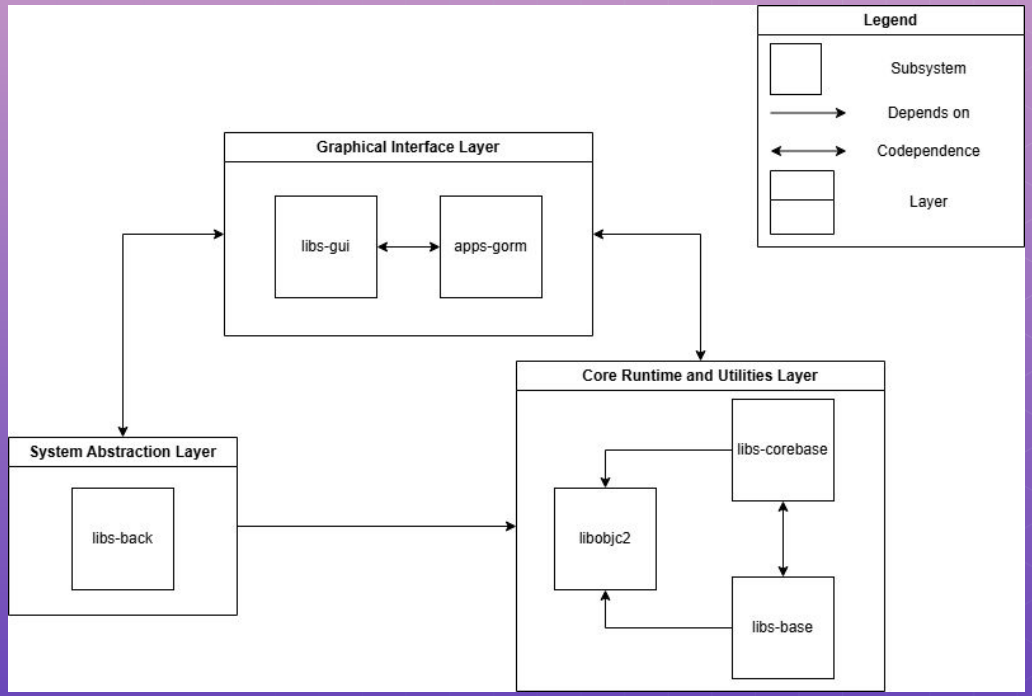
Key Features

Gorm Cloud allows users to upload templates, palettes, and GUIs to the cloud, either privately or publicly. Public uploads can be browsed and imported by other users, fostering collaboration and resource sharing.

Benefits & Integration

This feature reduces local storage needs and streamlines GUI development by providing easy access to shared designs. Integration requires minimal changes, with only the Gorm app connecting to a cloud server, while the rest of GNUStep remains unchanged.

Current State of The System



Existing Structure: Based on the layered architecture from the previous Concrete Architecture Report.

Key Benefits: Improved dependency management, modularity, and maintainability.

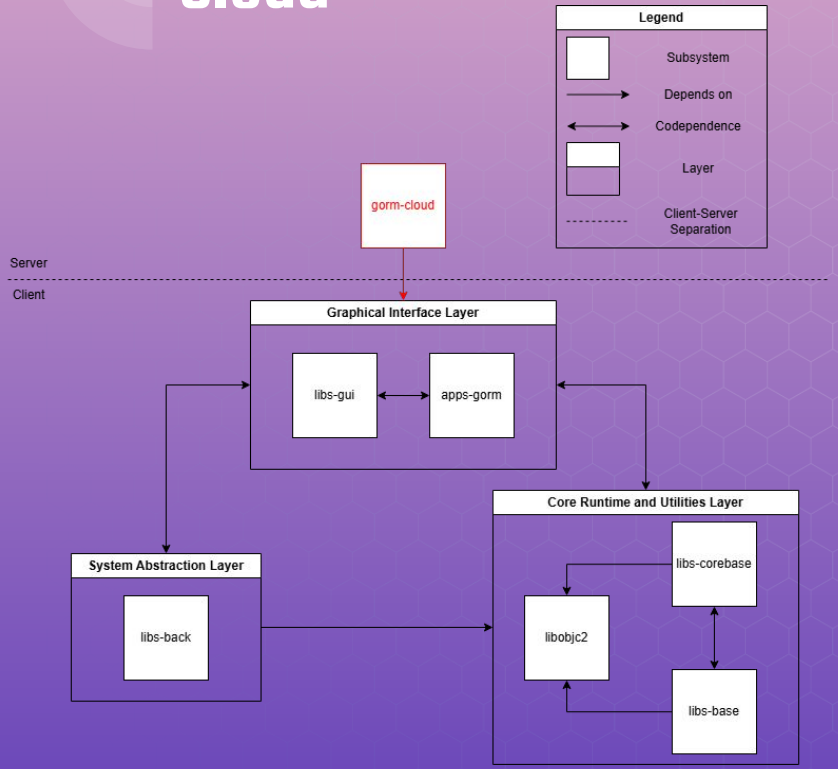
Next Steps: Modifying the architecture to integrate cloud capabilities for GNUStep.

SEI SAAM Analysis: Stakeholders and NFRs

Non-Functional Requirements (NFRs)

- **End-Users:** Usability, performance, and security.
- **Maintainers & Contributors:** Maintainability and scalability.
- **Org Teams:** Collaboration and cross-platform support.
- **Cloud Providers:** Interoperability and compliance.
- **Investors:** Cost efficiency and market competitiveness.
- **Development Team:** Testability and DevOps integration.

Implementation I:Client-Server Architecture for Gorm Cloud



Current System: The apps-gorm subsystem remains on the client-side.

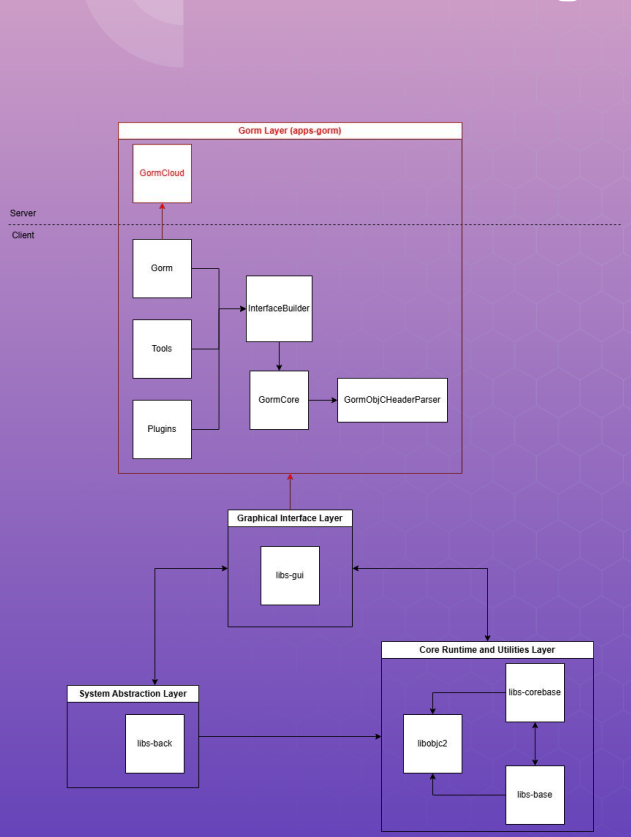
New Subsystem: gorm-cloud (server-side) manages a database for storing and retrieving templates, palettes, and GUIs.

New Component: CloudInterface is added to the apps-gorm subsystem to facilitate communication with gorm-cloud. It handles data uploads and retrievals.

Architecture Impact:

- Minimal changes to the original layered architecture.
- Client-side remains unchanged, with the server-side introducing the new subsystem.
- The client-server model ensures easy data distribution and scalability.

Implementation 2: Cloud-Based GNUStep with Advanced Cloud Manager



- **Structure Overview:** The second implementation integrates gorm into a cloud-based environment within the client-side apps-gorm subsystem, with enhanced server-side modules.
- **Key Enhancement:** Introduction of an **advanced cloud manager** with various **microservices** to handle local data storage, caching, and asynchronous synchronization.
- **Scalability & Performance:**
  - Components within the apps-gorm subsystem are independently scalable.
  - Improved scalability and reduced latency for accessing cloud data.
- **Unique Approach:** Focuses on increased **modularity** and **scalability**, optimizing performance for efficient cloud data access and management.

Comparison of Implementations

Feature	Implementation 1 (Client-Server)	Implementation 2 (Modular Cloud)
Architecture	Client-server model with a new gorm-cloud subsystem	Cloud-integrated apps-gorm with microservices
Data Storage	Centralized database in gorm-cloud	Distributed storage with local caching
Communication	Uses CloudInterface for data retrieval	Asynchronous synchronization for efficiency
Scalability	Server can be scaled separately	Modular structure allows independent scaling
Performance	Simple but may introduce latency	Optimized for low latency and fast access

Effects on Subsystems and Directories

High-Level Subsystems

Introduction of **gorm-cloud subsystem** to incorporate cloud storage

Shift towards **client-server architecture** (app-gorm on client side, gorm-cloud on server side)

Low-Level Subsystems

New CloudInterface component added to *apps-gorm* for cloud-based interactions

Integration with gorm-cloud introduces a new dependency

Cloud data will be processed and rendered like local data, keeping existing data handling largely unchanged

Directories and Files

New directories for **gorm-cloud** to support server-side cloud storage functions

CloudInterface files added to apps-gorm to manage client-server communication

Reorganization of file structure to separate cloud-stored assets and manage access controls (public/private)

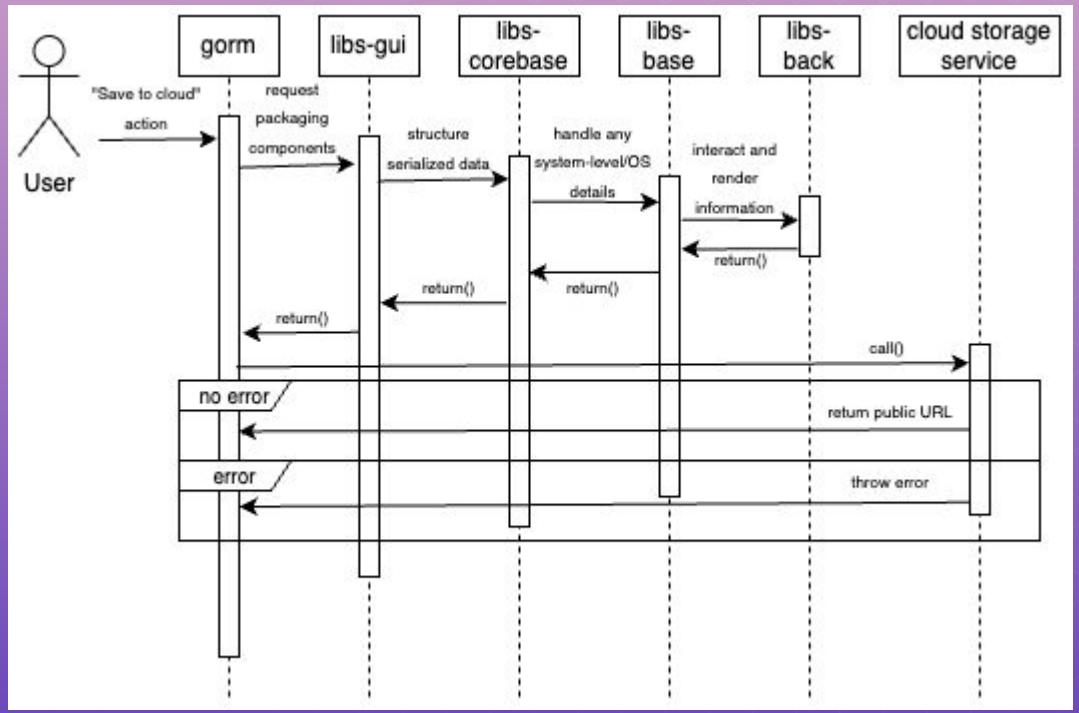
Potential Risks

- Increased complexity and maintenance challenges due to added subsystems and microservices
- System reliability risks—failure of one component can disrupt the entire system
- Network dependency introduces latency/bottleneck risks, impacting performance
- Greater architectural scope raises potential for overlooked issues during development

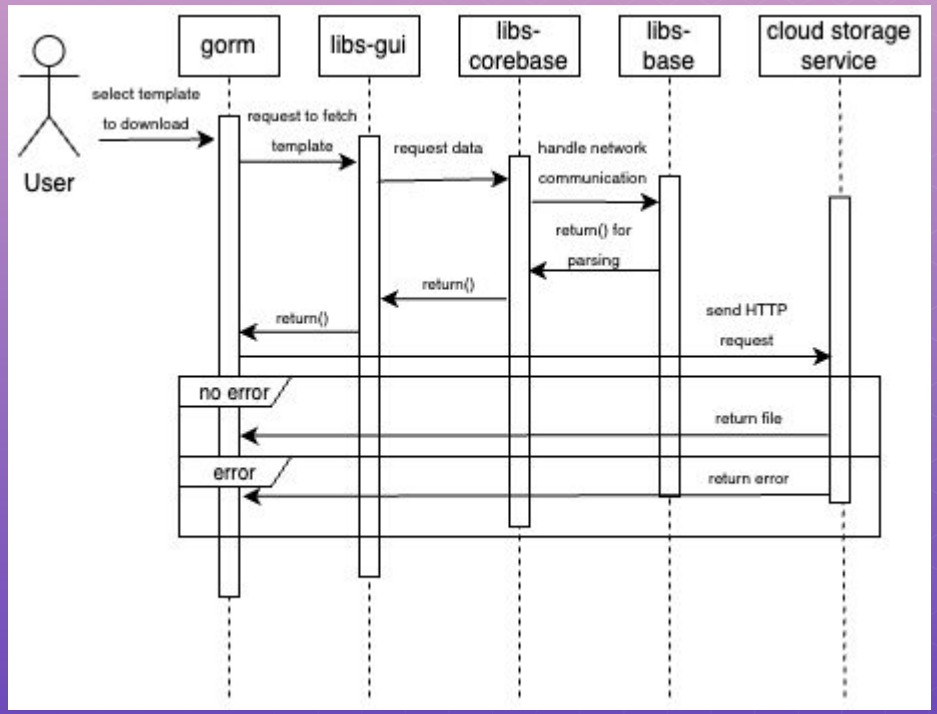
Testing

Unit Testing	Client-Server Communication	Performance Testing	Security Testing	User Experience Testing
Validate CloudInterface component for reliable uploads/downloads without data issues.	Test actions such as browsing, uploading and error conditions (e.g., network failure) to ensure synchronization and stability.	Measure latency, perform load & stress tests using testing tools to assess scalability.	Evaluate authentication and encryption mechanisms to protect user data.	Ensure UI is intuitive and user-friendly for uploading, downloading, and asset browsing.

Use Case I: User Saves a Custom GUI to the Cloud Publicly



Use Case 2: User Downloads a Custom Template from the Cloud



Lessons Learned

1. Gained deeper understanding of GNUStep’s architecture, including its strengths and weaknesses

2. Recognized both benefits and challenges of implementing Gorm Cloud (e.g., scalability vs. latency and security concerns)

3. Learned the value of thorough planning and analysis when proposing major system enhancements

Conclusion

Gorm Cloud seeks to:

Enables scalability and centralized cloud storage, allowing users to access and manage templates, palettes, and GUIs more efficiently

Improves collaboration and accessibility, making development available across multiple devices and environments

Modernizes GNUStep's architecture, aligning it with current client-server models

Thanks!

Questions?