https://youtu.be/6bVpLzd3rXA

A1: Conceptual Architecture

•••

Group leader: Spencer
Presenters: Tony, Aidan
Group members: Brant, Anson, Zhangzhengyang

Table of contents

- Overview
- System interactions
- Views
  - Functional
  - Information
  - Concurrency
- Architecture style
- Sequence diagrams

Table of contents

- ❏ Overview
- ❏ System interactions
- ❏ Views
  - ❏ Functional
  - ❏ Information
  - ❏ Concurrency
- ❏ Architecture style
- ❏ Sequence diagrams

Overview

- GNUstep is a free and open source object-oriented framework for developing desktop applications
- Implements the OpenStep specification
- Core libraries:
  - libs-back
  - libs-gui
  - libs-base
  - libs-corebase
- IDE:
  - Gorm

Codebase Overview

- Primarily written in Objective-C
- Stored in repositories under the GNUstep GitHub organization
- Core libraries:
  - Headers/ directory contains protocol declarations and class headers
  - Sources/ directory contains concrete protocol and class implementations
- Gorm:
  - Palettes are reside under the Applications/Gorm/ directory
  - Interface framework reside in InterfaceBuilder/ directory
  - Main interface construction facilities reside in GormCore/ directory

Table of contents

- Overview
- System interactions
- Views
    - Functional
    - Data and control flow
    - Concurrency
- Architecture style
- Sequence diagrams

System interactions

- Core libraries:
  - libs-base and libs-corebase are alternative implementations of FoundationKit
  - libs-gui and libs-back together implement AppKit
  - libs-gui defines graphical elements and auxiliary functions
  - libs-back handles drawing content to the display system
  - libs-gui and libs-back talk to each other through a DPS layer
- Gorm:
  - InterfaceBuilder defines abstract classes to support building interfaces
  - GormCore implement classes in Interfacebuilder
    - Inspectors handles connecting *outlets* and *actions*
    - Editors handle editing *controls*
  - Palettes under the main graphical application call into GormCore for their functionality

Table of contents

- Overview
- System interactions
- Views
    - Functional
    - Information
    - Concurrency
- Architecture style
- Sequence diagrams

Views - Functional

- Core libraries:
    - libs-back interacts with libs-gui via a DPS layer
    - libs-base provides key mechanisms for message passing and serialization
        - Notifications: asynchronous message exchange within a process
        - Remote connections: calling methods in remote processes
        - Archive: serialization of objects (and associated object) in text and binary formats
    - libs-gui provides event manager to handle publish/subscribing to peripheral events
- Gorm
    - Palettes interact with the current workspace via GormCore routines

Views - Information

- Core libraries:
  - libs-back and libs-gui exchange information through a static graphical context and display server context
  - libs-base and libs-corebase defines basic collection types
  - libs-corebase is interopable with libs-base via “toll-free bridging”
- Gorm:
  - Palettes do not hold any state about the current workspace
  - Delegate reads and writes to a corresponding controller in GormCore

Views - Concurrency

- Core libraries:
  - Multi-threading facilities:
    - NSThread
    - NSLock
    - NSOperation
    - NSOperationQueue
  - Asynchronous facilities:
    - NSNotification
    - NSNotificationCenter
  - RPC facilities
    - NSConnection
    - NSDistantObject

- Gorm
  - Central access to GormDocument through GormCore

Table of contents

- ❏ Overview
- ❏ System interactions
- ❏ Views
    - ❏ Functional
    - ❏ Information
    - ❏ Concurrency
- ❏ Architecture style
- ❏ Sequence diagrams

Architecture style

- GNUstep is split into two main architecture styles
  - Core libraries employ a layered architecture
  - Gorm uses a repository architecture

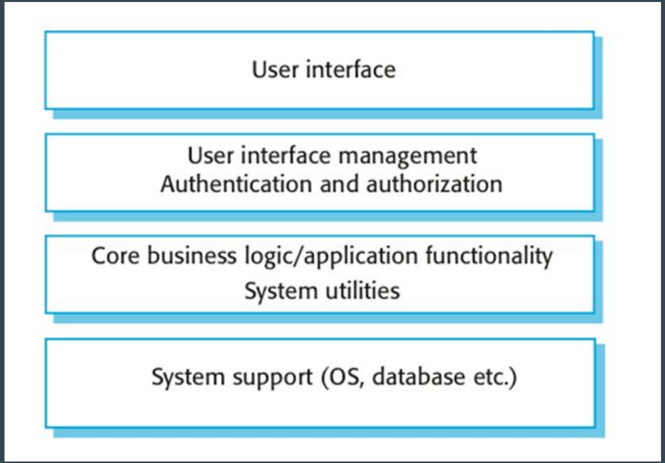
User interface
User interface management
Authentication and authorization
Core business logic/application functionality
System utilities
System support (OS, database etc.)

Figure 1. Layered architecture diagram [1]

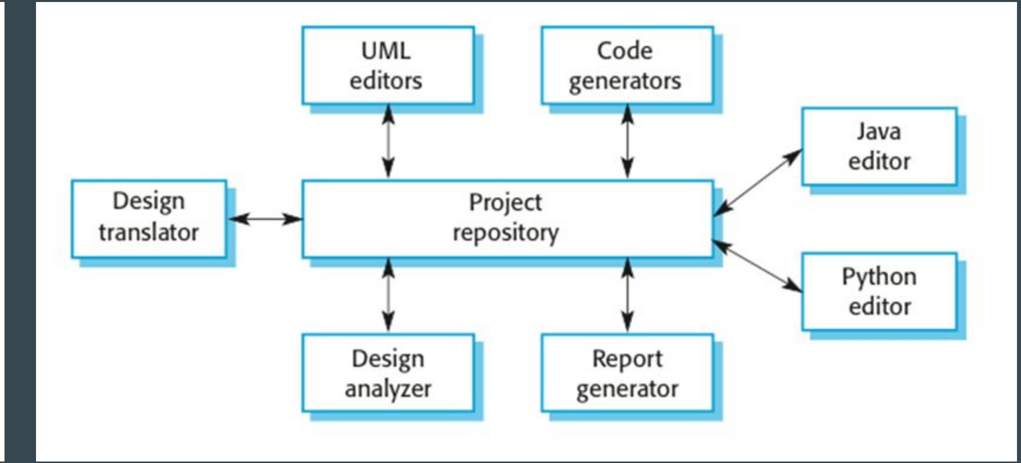

Figure 2. Repository architecture diagram [1]

Architecture style - core libraries

- ❑ Base layer (libs-base, libs-corebase):
    - ❑ Abstract over OS system calls
    - ❑ Provide facilities for networking/RPC and threading
    - ❑ Define basic data structures, protocols and models
- ❑ Application layer (libs-gui, libs-back):
    - ❑ Abstract over display operations
    - ❑ Provides graphical element primitives and auxiliary functions
    - ❑ Built on top of data structures, protocols and models defined in the base layer

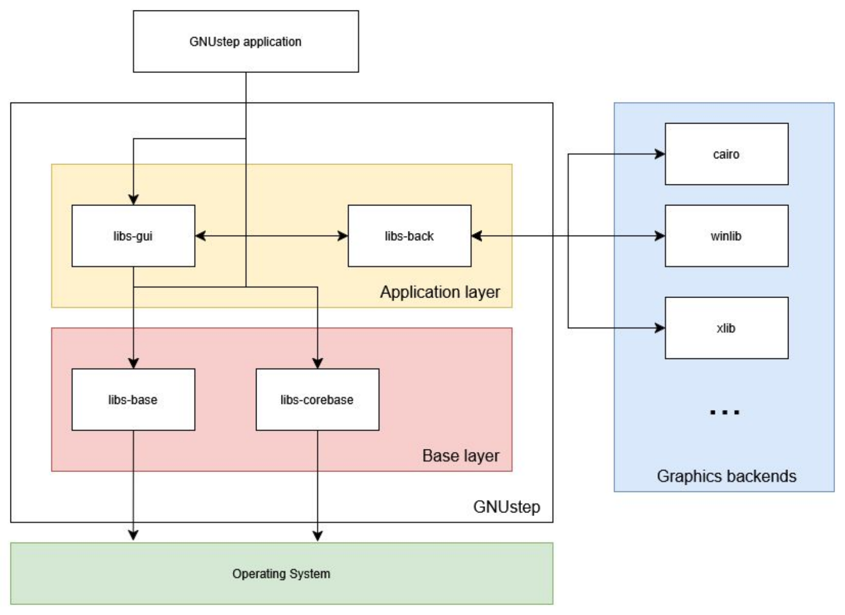

Figure 3. Core libraries architecture

Architecture style - Gorm

- Repository style
  - Palettes share access to the working document and call into GormCore to perform read/writes
  - Individual units independently update a shared document

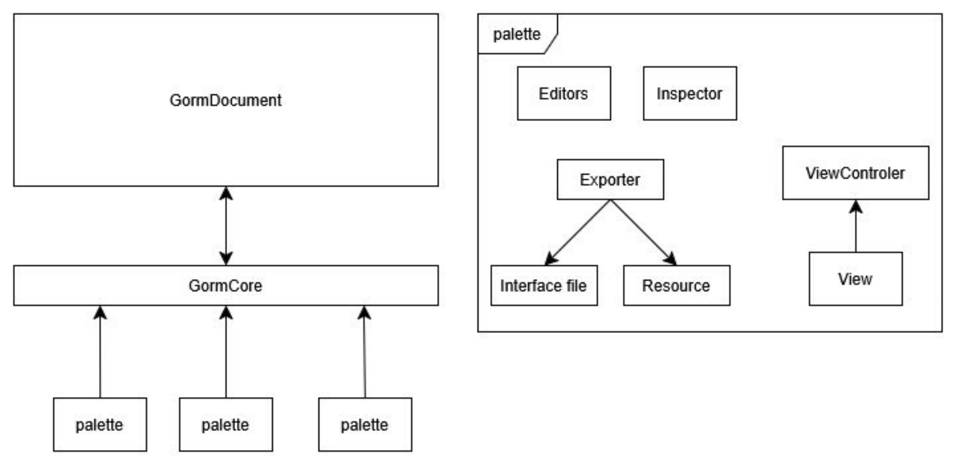

Figure 4. Gorm architecture

Table of contents

- Overview
- System interactions
- Views
  - Functional
  - Information
  - Concurrency
- Architecture style
- Sequence diagrams

Sequence diagrams (legend)

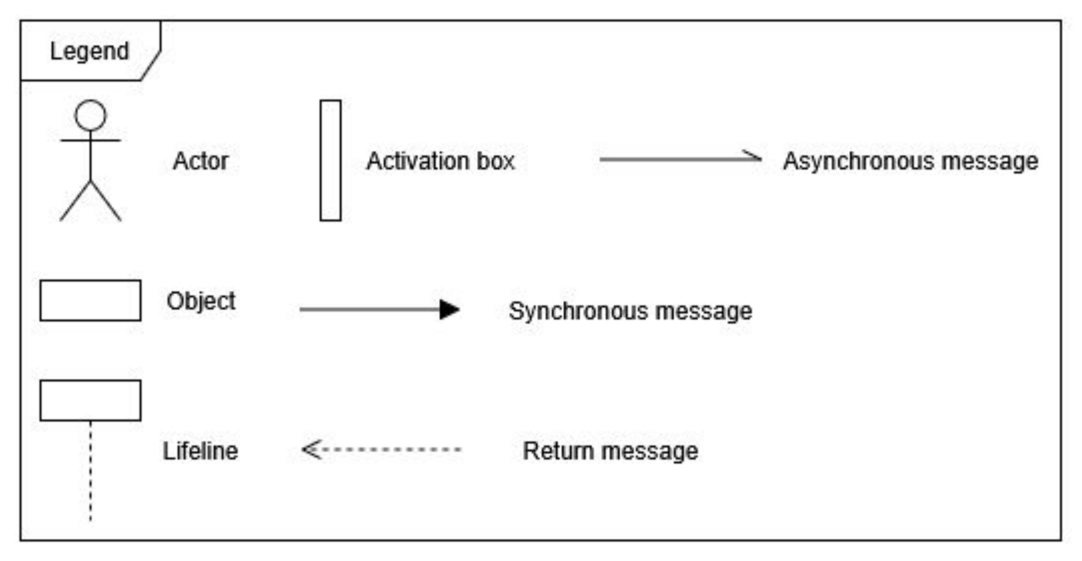

Figure 5. Sequence diagram legend

Sequence diagrams - core libraries

Background: “GuessTheNumber'” is an application generates a secret random number between 0 and 256 and presents a graphical form with to user asking for an integer between 0 and 256 along with a button to submit the form. Upon submission of the form, the program print whether the user input matches the secret number to screen and clears the form.

Use case: User inputs a number in “GuessTheNumber” and then clicks on the button to submit the number.

Sequence diagrams - core libraries

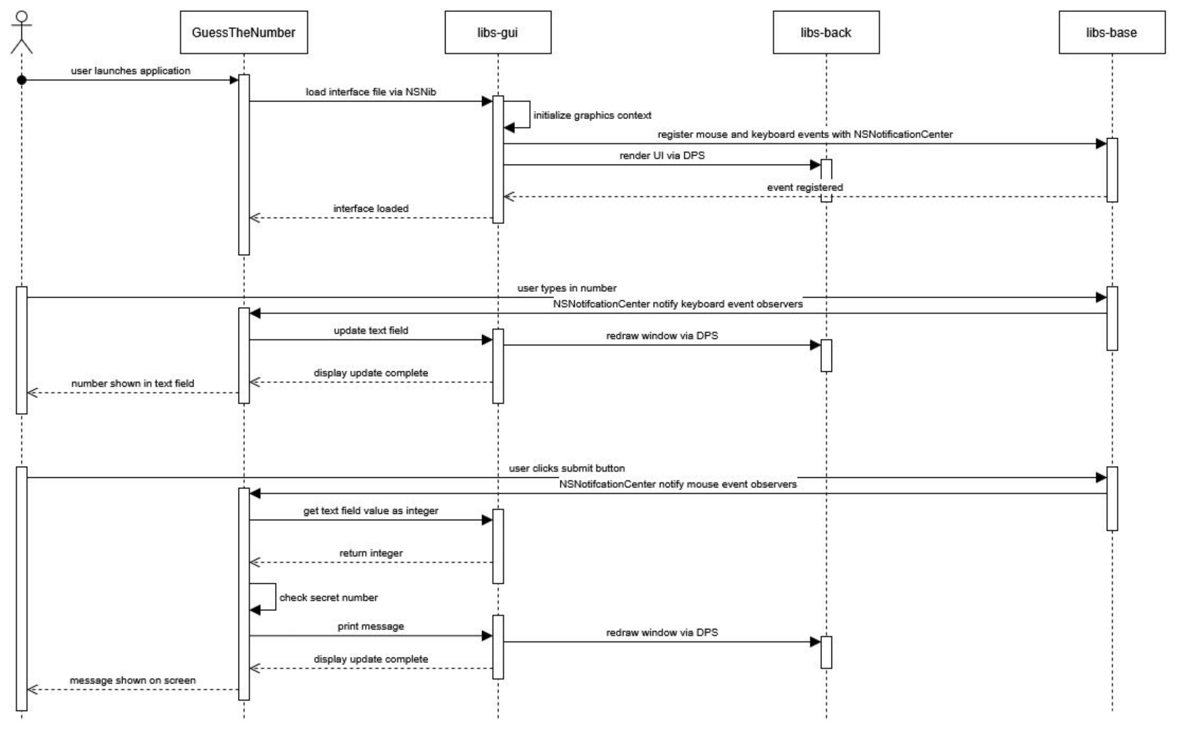
GuessTheNumber
libs-gui
libs-back
libs-base
user launches application
load interface file via NSNib
initialize graphics context
register mouse and keyboard events with NSNotificationCenter
render UI via DPS
event registered
interface loaded
user types in number
NSNotifcationCenter notify keyboard event observers
update text field
redraw window via DPS
display update complete
number shown in text field
user clicks submit button
NSNotifcationCenter notify mouse event observers
get text field value as integer
return integer
check secret number
print message
redraw window via DPS
display update complete
message shown on screen

Figure 6. Sequence diagram 1

Sequence diagrams - Gorm

Background: The developer launches Gorm and opens a new workspace. For simplicity, we disregard any calls to libraries outside of Gorm, for instance to libs-gui for rendering the graphical editor.

Use case: Developer use the Windows palette to drop a window into the workspace and use the Inspector palette to rename the window title to "File Viewer". The interface layout is then saved to the disk.

Sequence diagrams - Gorm

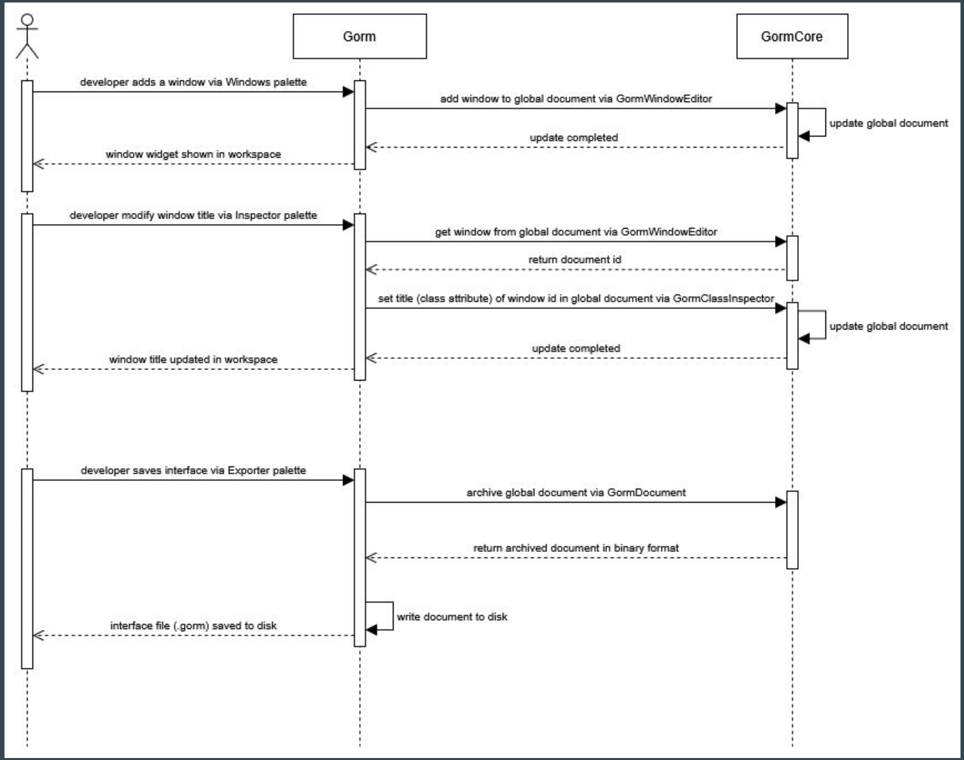
Gorm
GormCore
developer adds a window via Windows palette
add window to global document via GormWindowEditor
update global document
update completed
window widget shown in workspace
developer modify window title via Inspector palette
get window from global document via GormWindowEditor
return document id
set title (class attribute) of window id in global document via GormClassInspector
update global document
update completed
window title updated in workspace
developer saves interface via Exporter palette
archive global document via GormDocument
return archived document in binary format
write document to disk
interface file (.gorm) saved to disk

Figure 7. Sequence diagram 2

Image credits

1. https://csis.pace.edu/~marchese/SE616_New/L6/L6.htm

Thank You