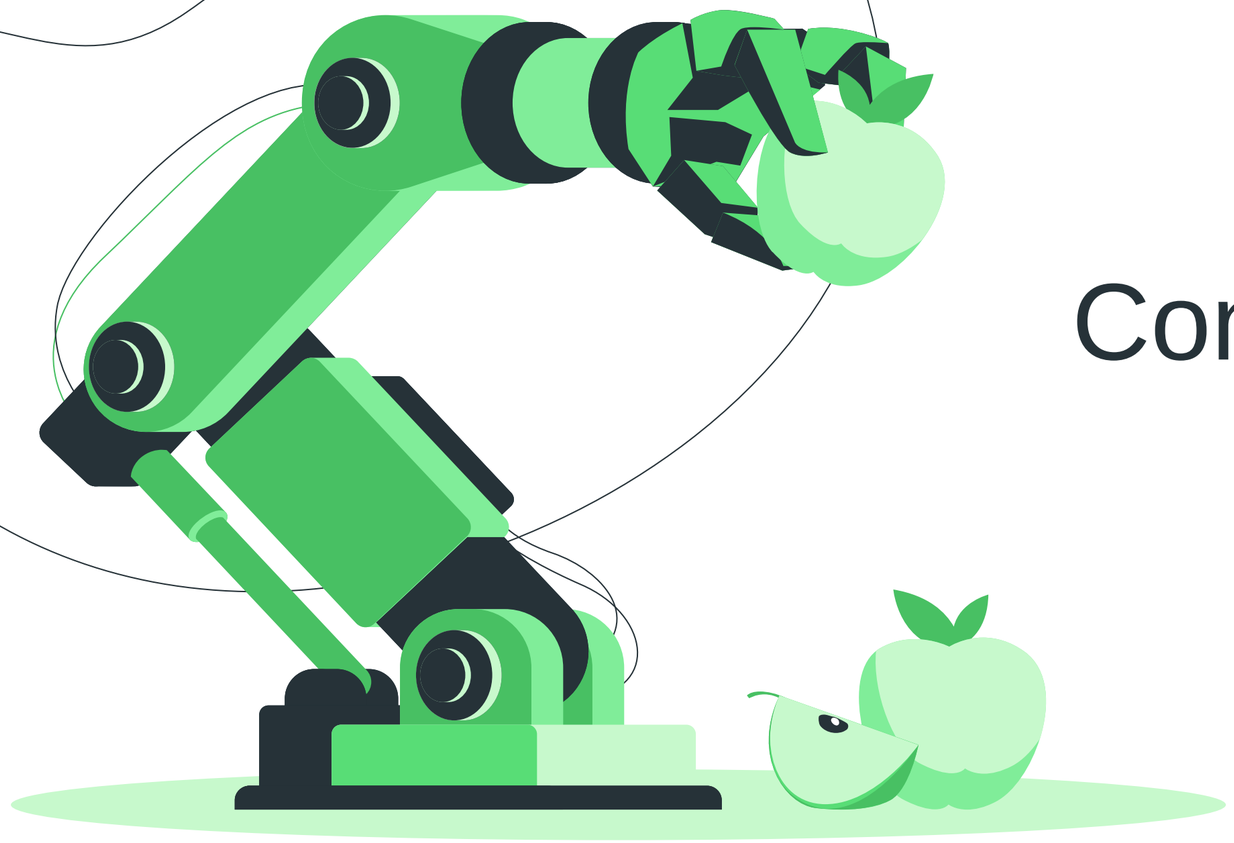


GNUstep Architecture

Conceptual vs Concrete

Presented by: Kiarash Mirkamandari
and Ebrahim Haghshenas

Agenda

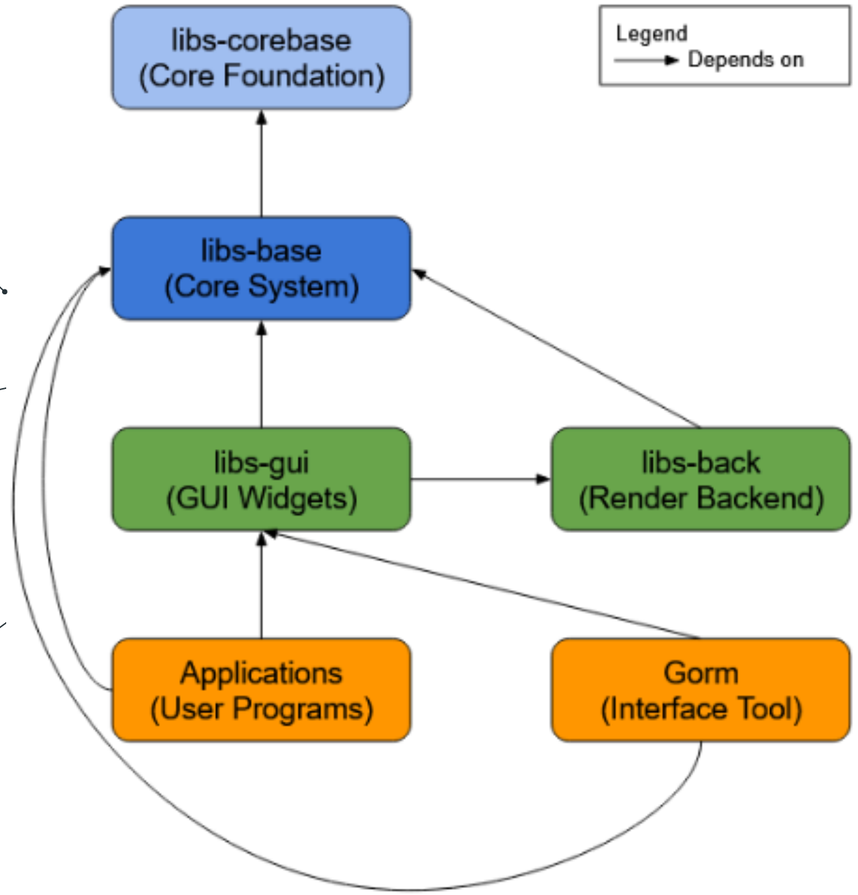
1. Conceptual Architecture
2. Concrete Architecture & Derivation
3. Subsystem Deep-Dive
4. Sequence Diagram
5. Concurrency & Team Issues
6. Reflexion Analysis (Difference + Rationale)
7. Lessons Learned

Conceptual Architeture Overview

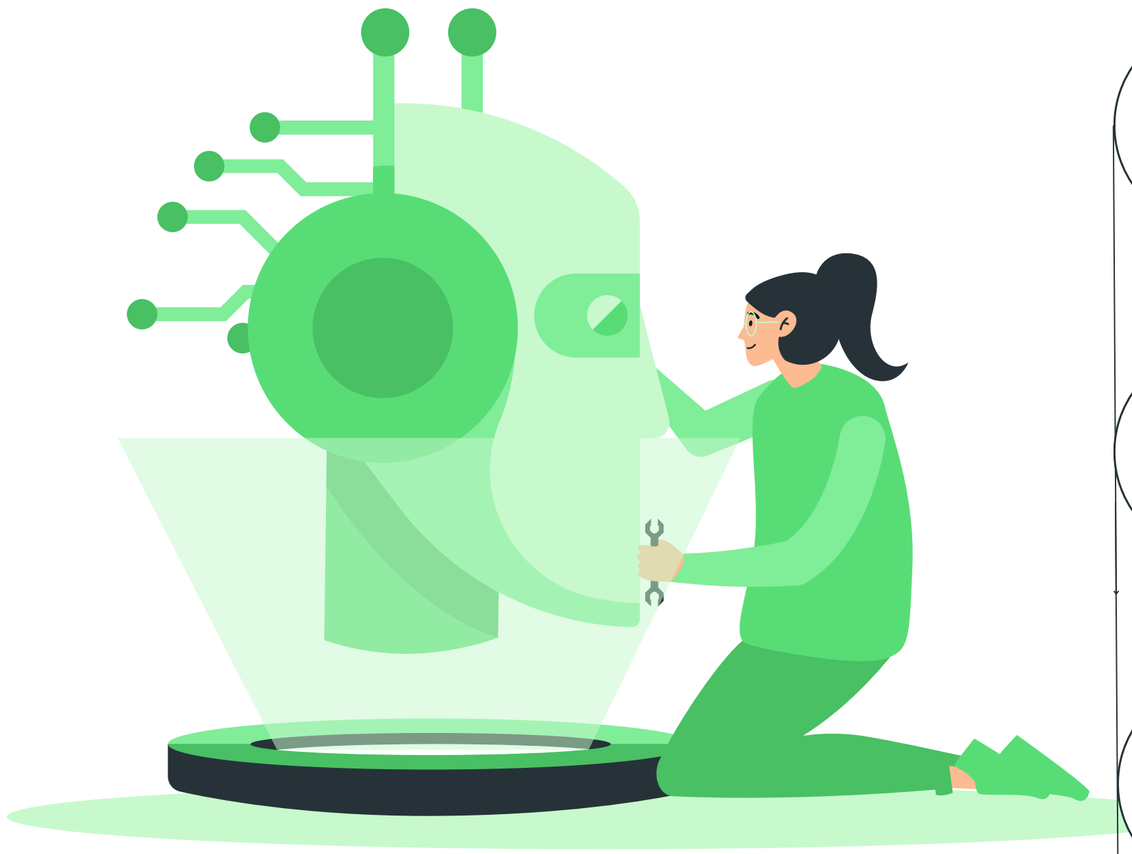
GNUstep is an open-source implementation of Apple's OpenStep/Cocoa

Key conceptual layers: libs-corebase, libs-base, libs-gui, libs-back, libobjc2.

Originally assumed strict layering and single backend in A1.



Updated Conceptual Architecture & Rationale



libobjc2 separated from libs-base to show the modern Objective-C runtime.

libs-back split into multiple backend modules (X11, Win32, etc.).

Partial or bypassed libs-corebase due to missing features.

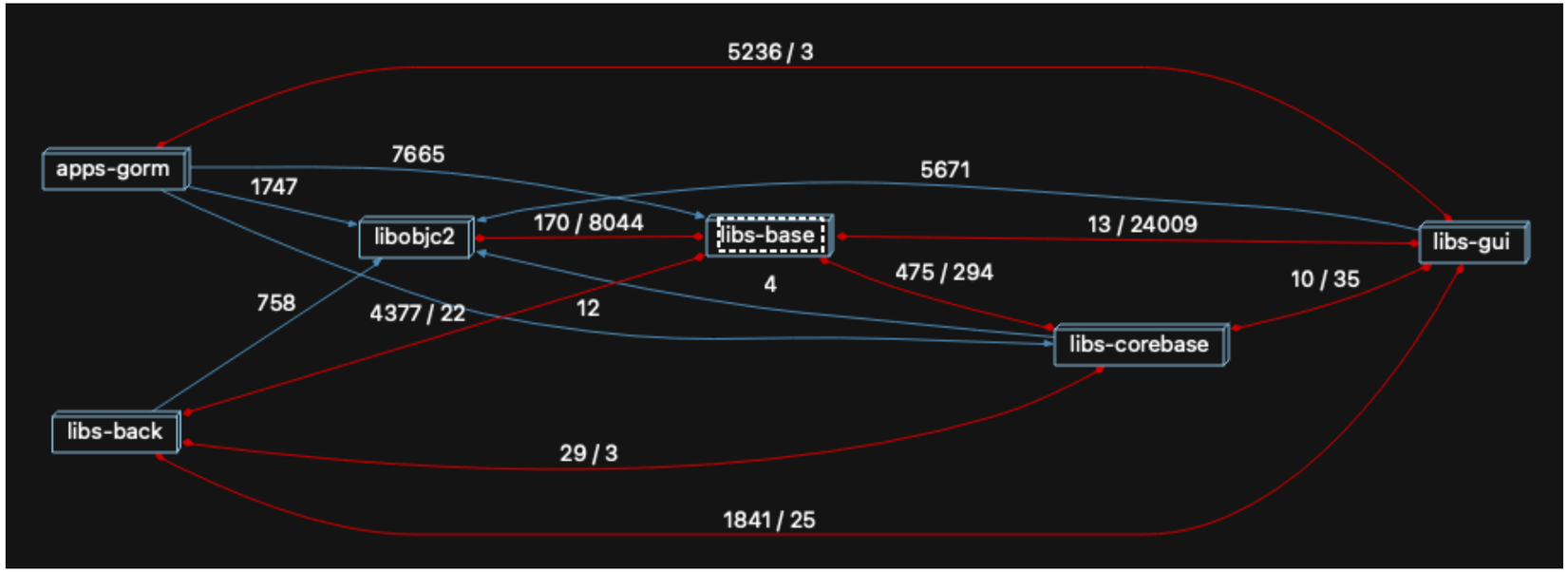
Concrete Architecture & Derivation

Used Understand + manual inspection for actual dependencies.

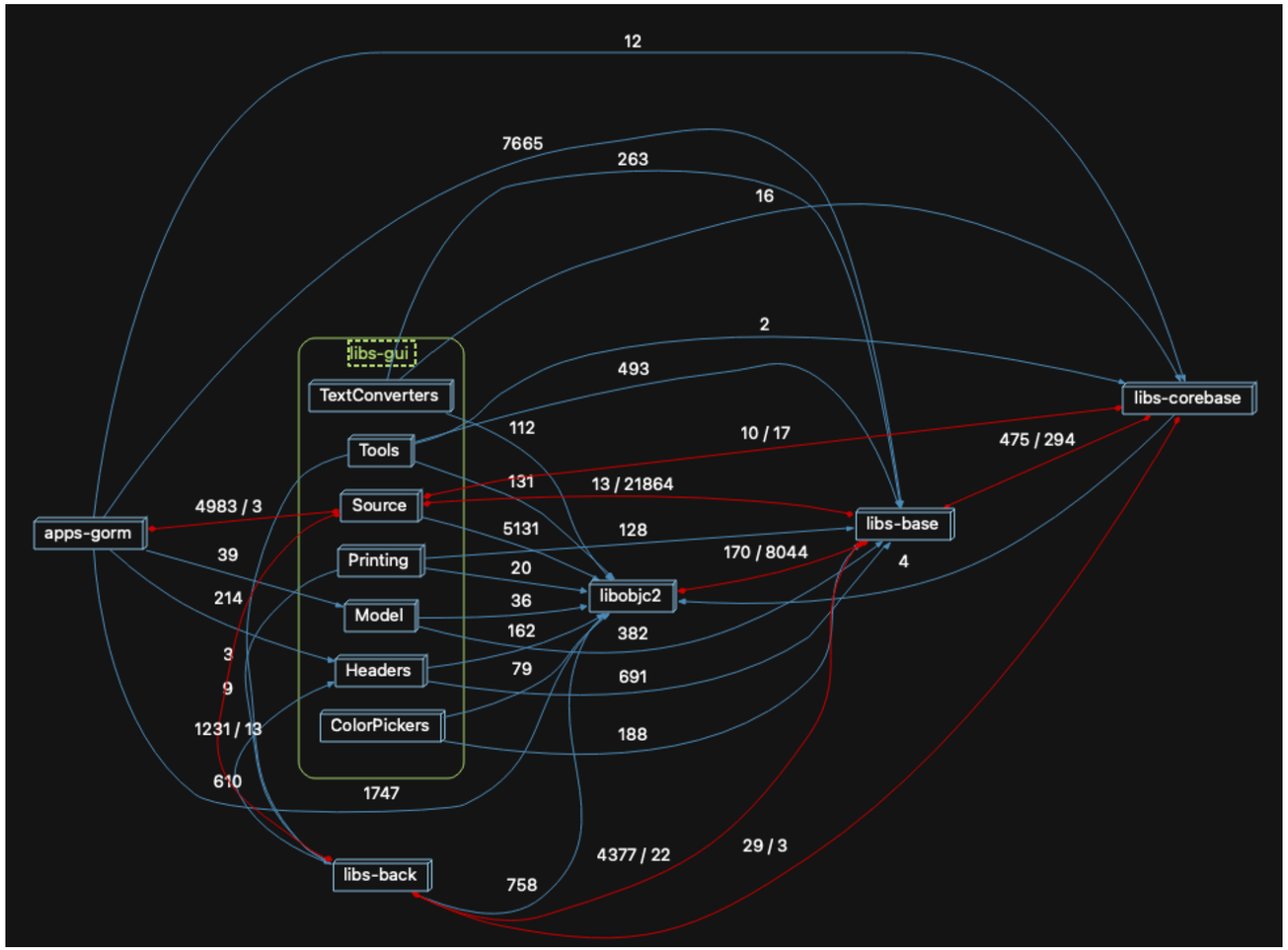
Grouped source files by functionality into recognized subsystems.

Mapped observed calls back to conceptual design.

Considered alternative groupings for OS-level functions.

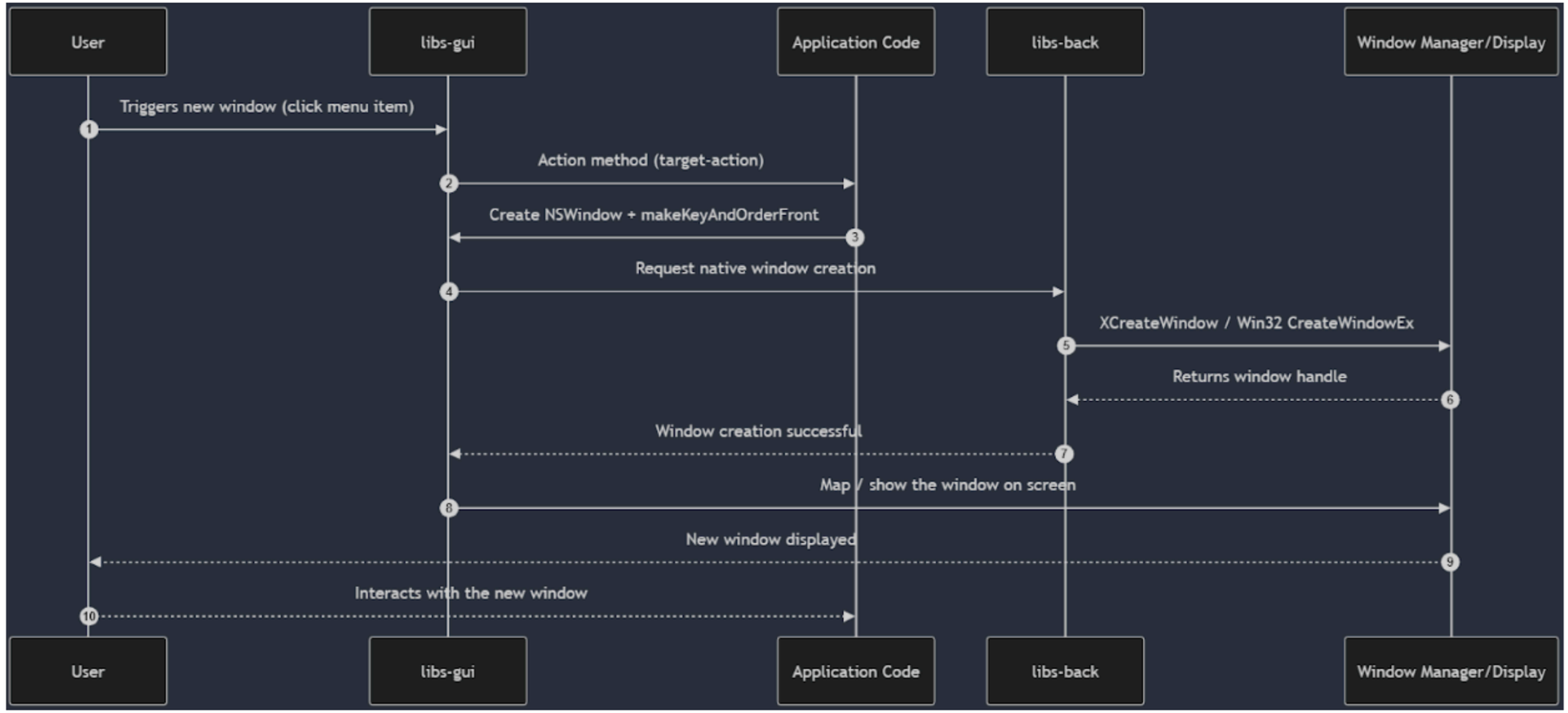


Subsystem Deep Dive: libs-gui



1. Business logic/data (libs-base)
2. Relies on libs-base for concurrency and data structures.
3. Delegates drawing to libs-back for platform-specific rendering.

Subsystems & Components

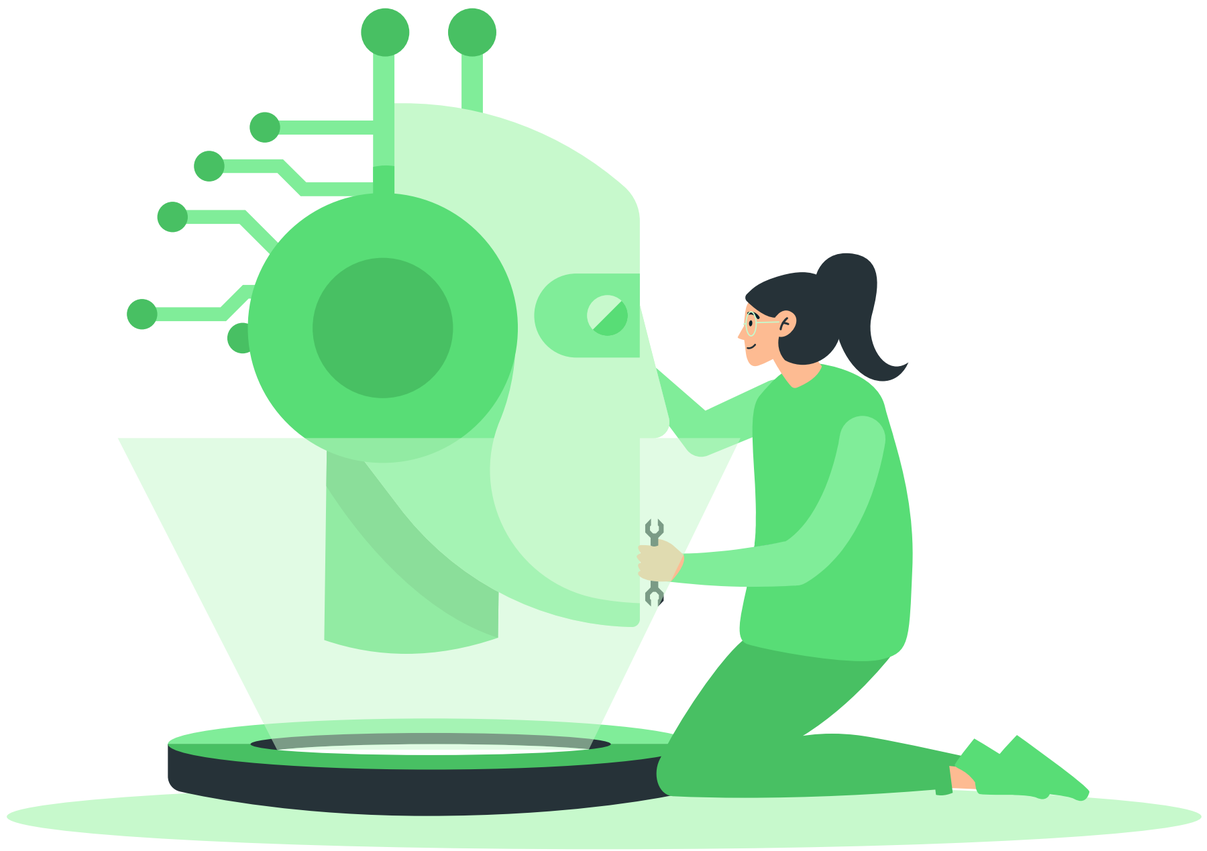


1. User clicks 'New Window' → libs-back captures event
2. libs-gui calls target-action on application controller.
3. NSWindow object is created and configured in libs-gui.
4. libs-back invokes OS APIs (e.g. XCreateWindow) to display.

Concurrency & Team Issues

Main Thread	Team Roles	Performance Conflicts
• Single main thread for GUI, optional background threads from libs-base.	• Team splits tasks: base, GUI, and backend maintainers.	• Bypassing libs-corebase for performance can cause merge conflicts.

Reflexion Analysis & Differences



- libobjc2 is fully separate; initially lumped with libs-base.
- Multiple dynamic backends vs. single monolithic ‘libs-back.’
- Direct OS calls in libs-base instead of going through libs-corebase.
- Reasons: performance optimizations, incomplete libs-corebase, cross-platform needs.

Lessons & Limitations

1. Real code diverges from neat conceptual layers; shortcuts are common.
2. Dynamic backends add flexibility but increase complexity.
3. Overall: layered architecture remains, adapted for practicality.
4. Improving libs-corebase could reduce direct OS calls.