CONCRETE ARCHITECTURE OF GNUSTEP

Group# : 500-Internal Server Error

https://youtu.be/iQmk1TBe8bE

GROUP MEMBERS

Wanting Huang - 20wh18@queensu.ca (Group Leader)

Qiu Xing (Nathan) Cai - 21qxc@queensuca (Presenter)

Haoxi Yang - 21hy11@queensu.ca (Presenter)

Yuda Hu - 21yh29@queensu.ca

Nicholas Wang - 22nw3@queensu.ca

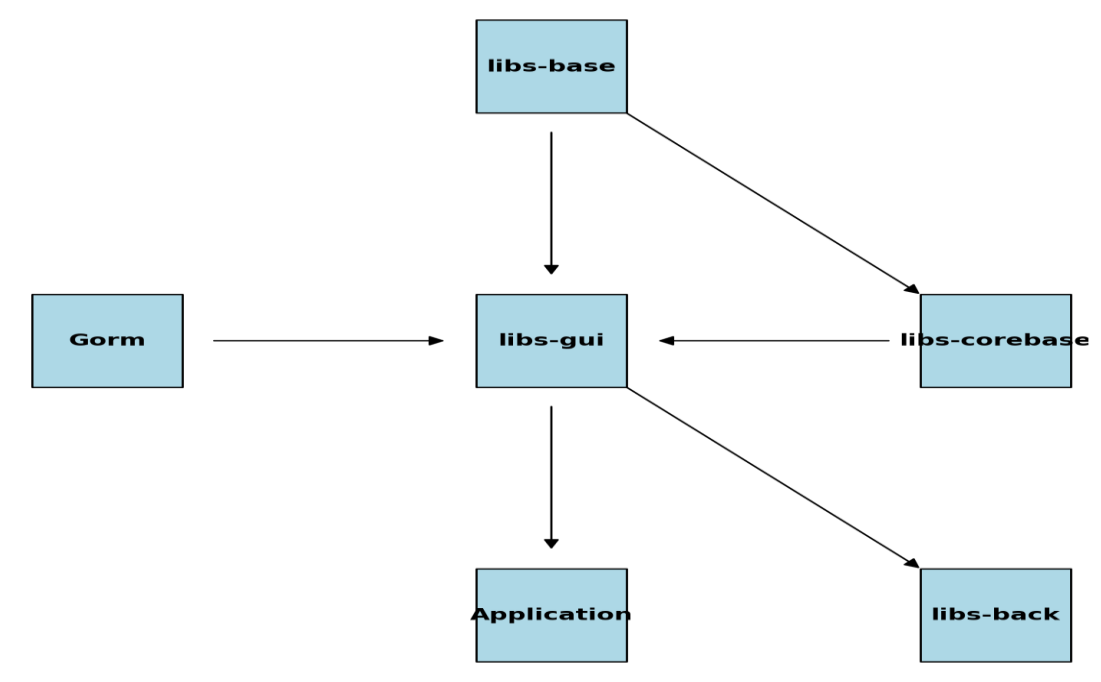
Amethyst Shen - 21YC121@QueensU.ca

CONTENT

INTRODUCTION

- Project goal is to analyze GNUstep's concrete architecture and compare it with its conceptual design
- Using SciTools Understand
- Compare Conceptual and Concrete Architecture
- Understand importance of real-world software architecture
- Identify potential design and potential improvement

UPDATED CONCEPTUAL ARCHITECTURE



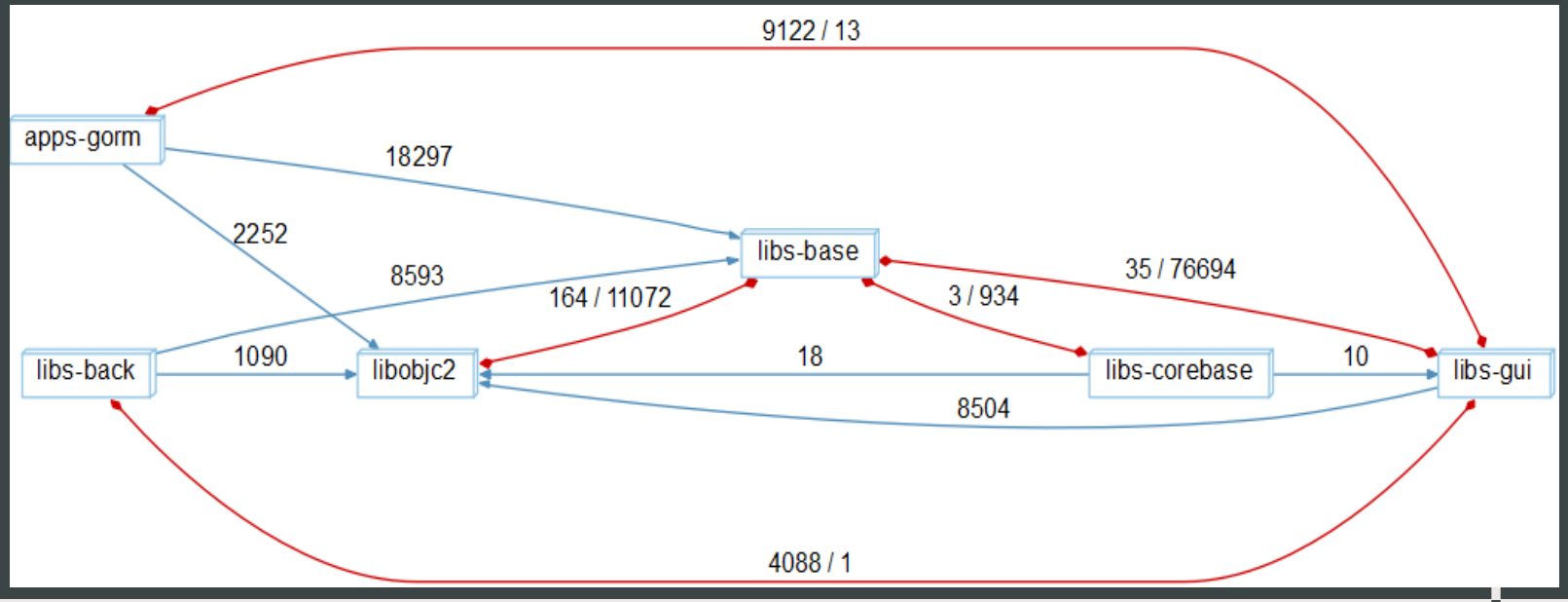
Key subsystems

- Libs-base
- Libs-corebase
- Libs-gui
- Libs-back
- Apps-gorm
- libobjct2

Architecture

- Layered Architecture
- Objected-Oriented Design
- Model-View-Controller

DERIVATION PROCESS



Methodology

- SciTools Understand
- Analyze dependencies
- Grouped Subsystems

Steps Taken

- Load GNUstep project on SciTools Understand
- Dependency graphs
- Grouped related components into functional subsystmes
- Compare

TOP-LEVEL CONCRETE ARCHITECTURE

concrete subsystems

- **Core Libraries**: libs-base, libs-corebase
- **GUI libraries**: libs-gui, apps-gorm
- **Rednering:** libs-back
- **Runtime Support**: libobjct2

Interaction

- libs-gui depends on libs-base for application logic
- Rendering handled by libs-back
- Apps-form interatis with libs-gui for UI design.
- Unexpected couplings: UI operations by pass libs-gui and interact directly with libs-base

ALTERNATIVE ARCHITECTURE STYLES

Layered Architecture

- Provides a balance of maintainability and efficiency
- Ensures celar seperation of concerns
- Allow tight integration for performance optimization

Microkernel Architecture

- Incresed modularity, better isolation
- Performance overhaed due to excessive inter-process communication

Component-Based Architecture

- Easier updates, better reusability
- Increased complexity in managing dependencies

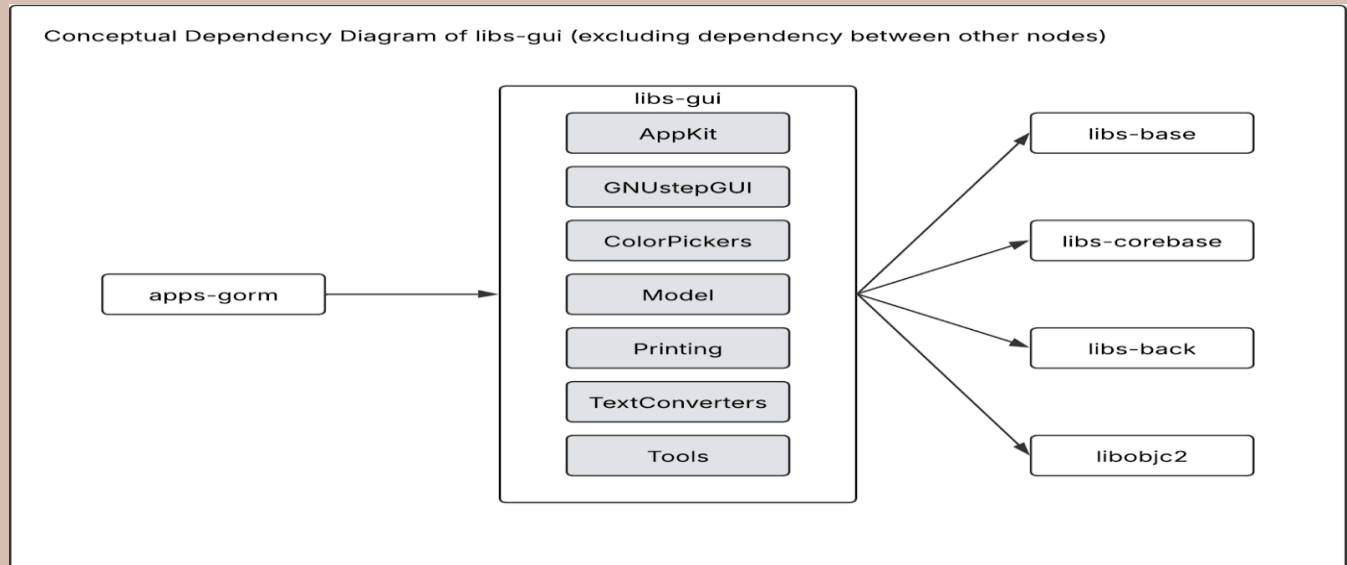
Service-Oriented Architecture (SOA)

- Independent services enhace reusability
- Higher communicaiton overhead and complexity

SUBSYSTEM ANALYSIS – CONCEPTUAL VIEW

For libs-gui

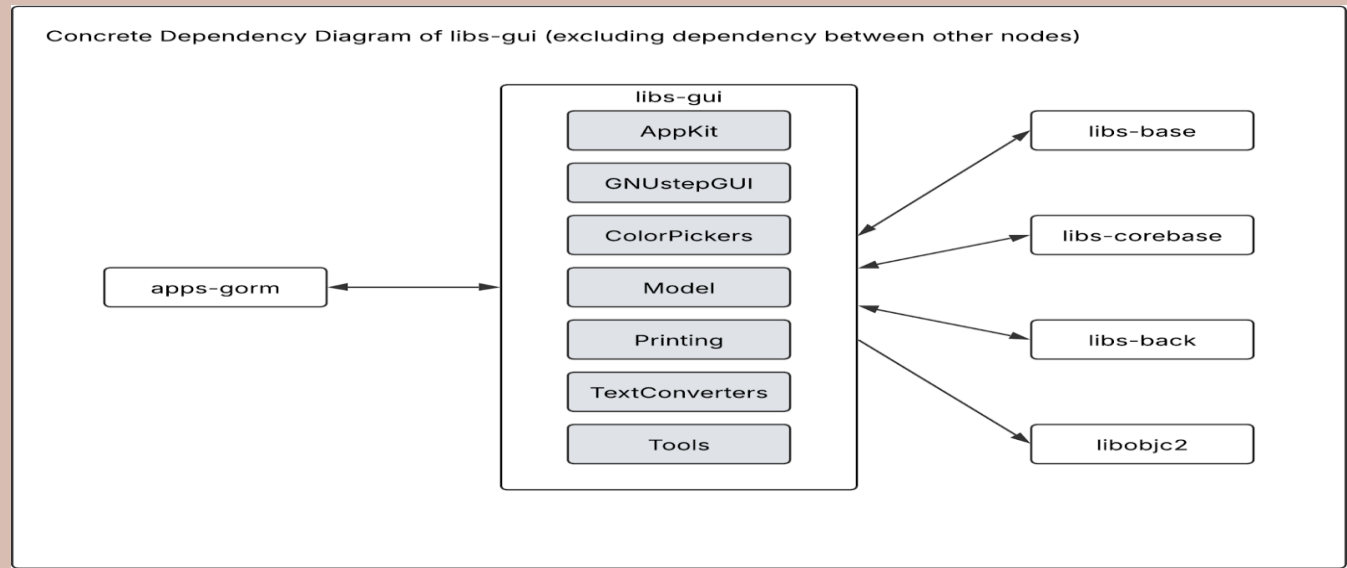
- libs-gui is divided into two parts, the frontend component which handles the GUI independent of platform and display system logic
- It depends on libs-back as the backend that handles the platform and display system, such as calls to the OS
- Apps-gorm is dependent on libs-gui to create the GUI in a drag-and-drop environment
- Libs-gui also relies on core dependencies from libs-base and libs-corebase



SUBSYSTEM ANALYSIS – CONCRETE VIEW

For libs-gui

- There are no absences from the conceptual view in the concrete view
- But there are some divergencies, where there were supposed to be uni-directional dependencies there are, in reality, bi-directional dependencies shared between:
  - Apps-gorm
  - Libs-base
  - Libs-corebase



REFLEXION ANALYSIS FOR 2ND LVL SUBSYSTEM

libs-base → libs-gui
One example is in the file libs-base/Source/NSMessagePortNameServer.m in lines 522 and 626, libs-base retrieves the port name through the NSMessagePort class's property several times which is from libs-gui/Source
libs-gui → apps-gorm
libs-gui overrides some of the methods of apps-gorm/GormCore such as toolbarSelectableItemIdentifiers in libs-gui/Source/NSTabViewController.m
libs-corebase → libs-gui
libs-corebase includes the config.h file several times from libs-gui/Source such as in libs/corebase/Source/CFRunloop.c

HIGH LEVEL REFLEXION ANALYSIS

There are some divergences we have observed, for example, libs-back should be dependent on libs-base and libs-corebase but it turns out those two subsystems also depend on libs-back even though those two subsystems provide the core dependencies for all of GNUStep.

libs-corebase→libs-back

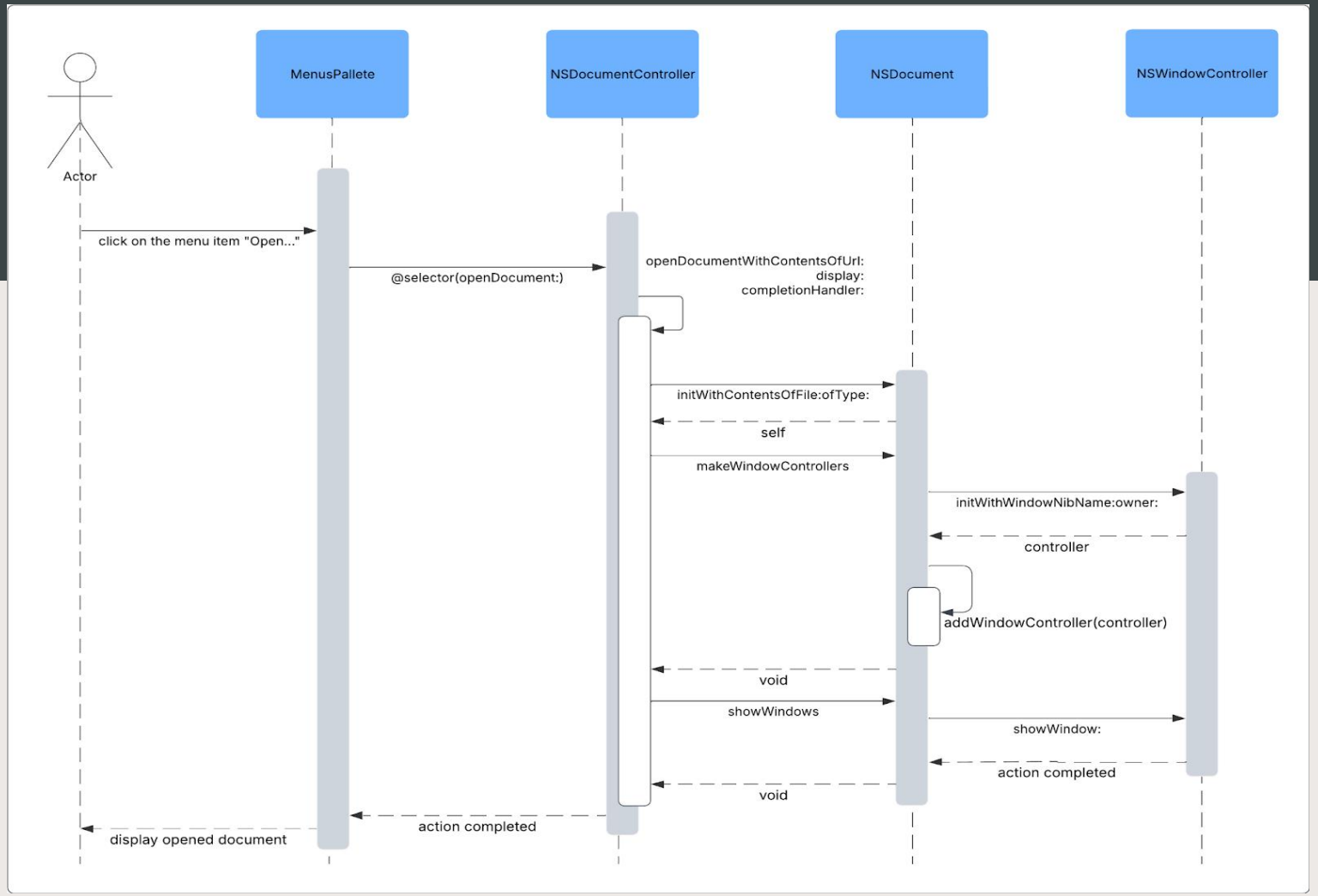
There is a special macro for TRUE and FALSE under libs-back/source/xdps/parseAFM.c that's use in libs-corebase as an equivalent for boolean data types like the return type for the function CFGregorianDateIsValid in libs-corebase/Source/CFDate.c

libs-base→libs-back

The boolean macro is also used in libs-base like it is in libs-corebase like for libs-base/Source/libgnustep-base-entry.m using it on line 62.

We believe for situations like this where there's a macro that serves as the boolean data type, which will likely be used everywhere, it would be a better design choice to put it under libs-base or libs-corebase since it is such a core functionalit

USE CASE: MENU ACTION EXECUTION



LESSONS LEARNED AND LIMITATIONS

Lessons Learned:

- We learned that all software projects don't go as planned, what was supposed to be a one-way dependency was often two-ways
- Looking into the code made us realize how big and complex the project is to understand
- How much GNUStep relies on the native system's tooling such as compilers, especially engines

Limitations:

- We could not always get a clear understanding of why there was a divergence in some dependencies imported as we do know the original developer nor PR that implemented it and why the dependency is located and imported the way it is

CONCLUSION

- In this assignment we got a deeper understanding of the underlying code of the GNUStep project
- We have explored the concrete view finding several discrepancies in our reflexion analysis when compared against the conceptual architecture in our first assignment
- We have adjusted our conceptual architecture and sequence diagrams to properly adjust to our deeper understanding of the project

THE END