GNUSTEP CONCRETE ARCHITECTURE

Group 12

CISC/CMPE 322/326

Link to video: https://youtu.be/kfHJd4vxI_E

GROUP MEMBERS

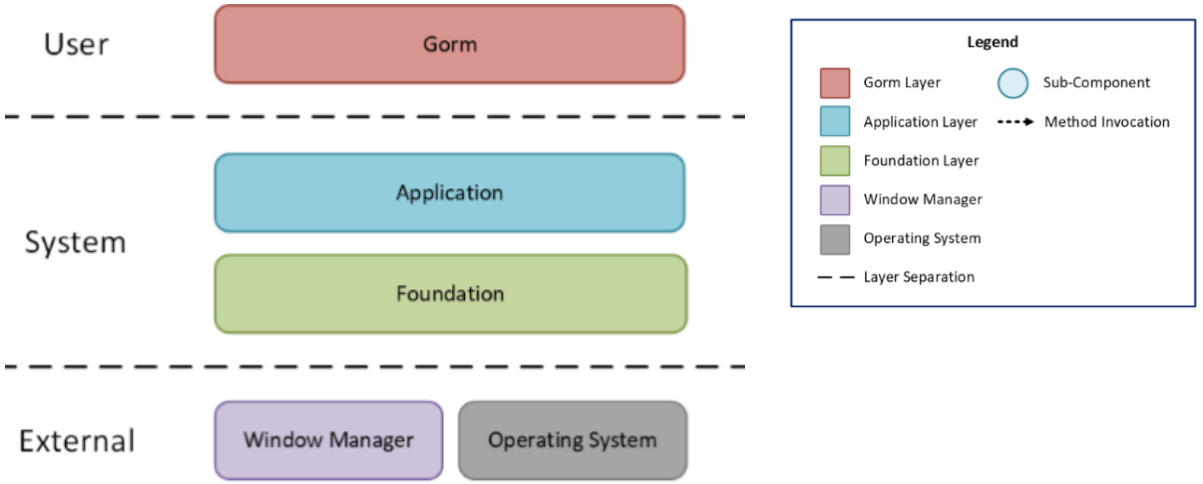
- Daniel Tian (Presenter): 21dt41@queensu.ca
- Samuel Tian (Presenter): 21st114@queensu.ca
- James Choi: 19jc132@queensu.ca
- Christian Pierobon: christian.pierobon@queensu.ca
- Luca Spermezan: 22ls18@queensu.ca
- Andrew Bissada (Leader): 21ajb37@queensu.ca

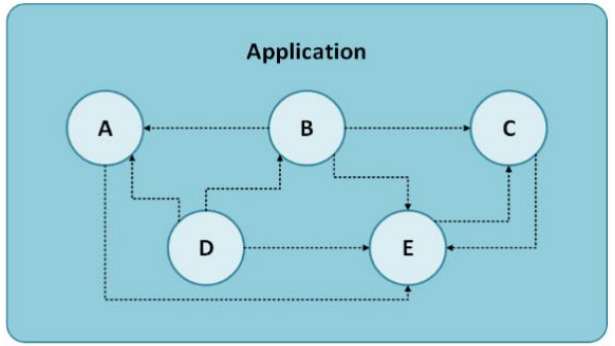
DERIVATION PROCESS

- **Phase 1:** Making revisions to conceptual architecture
- **Phase 2:** Sorting source code files from libraries to components
    - Consulted documentation for concrete classes on **GNUstep/Apple Developer** websites
- **Phase 3:** Revising dependencies for more accurate architecture

ARCHITECTURAL STYLES

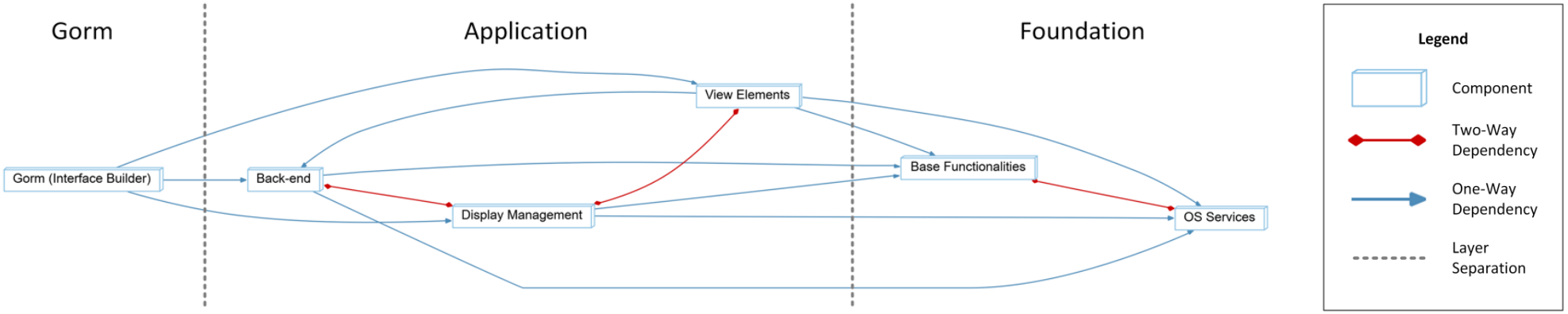
- **Layered:**
  - Foundation: libs-base, libs-corebase
  - Application: libs-gui, libs-back
  - Gorm: apps-gorm
- **Object-oriented:**
  - Classes used to define objects
  - Communication via method invocations





OVERVIEW OF CONCEPTUAL ARCHITECTURE

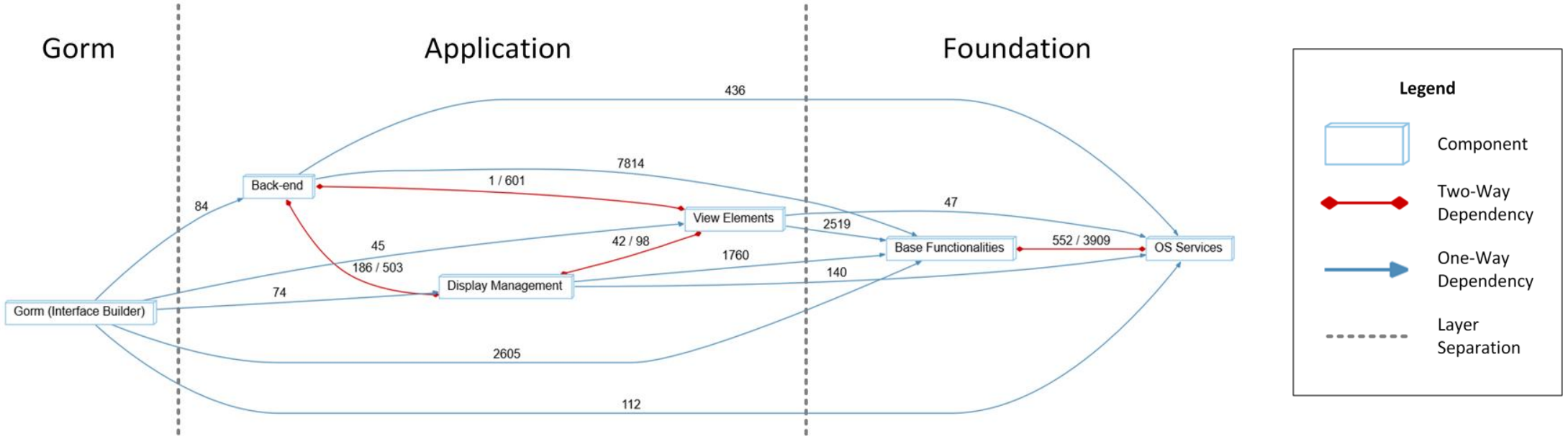
- **Aggregated sub-components:** Drawing, Event Handling, Notifications
- **Added top-level components:** Base Functionalities, Display Management
- **Added sub-components:** Runtime Utilities, File Data



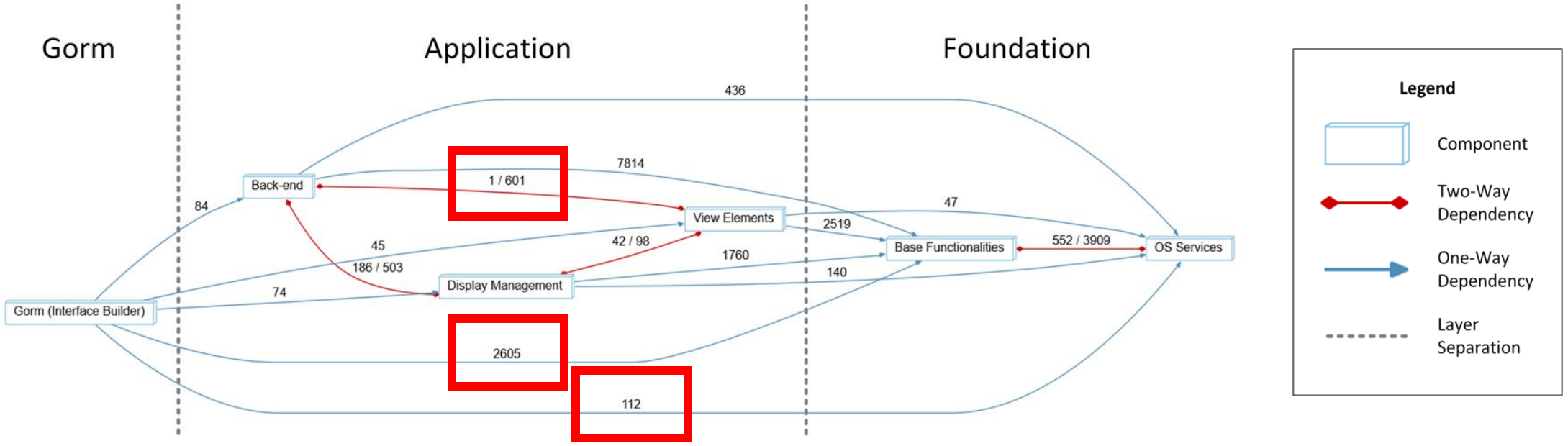
OVERVIEW OF CONCEPTUAL ARCHITECTURE

- **Foundation:**
  - **Base Functionalities:** Notifications, Runtime Utilities, Serialization and Archiving, Value Data
  - **OS Services:** File Management, Networking, Task Management
- **Application:**
  - **View Elements**
  - **Display Management**
  - **Back-end:** Main Application Environment, Event Handling, Drawing, Text, Audio, File Data
- **Gorm:**
  - **Gorm (Interface Builder):** Gorm UI, Main Gorm Application (Core)

OVERVIEW OF CONCRETE ARCHITECTURE



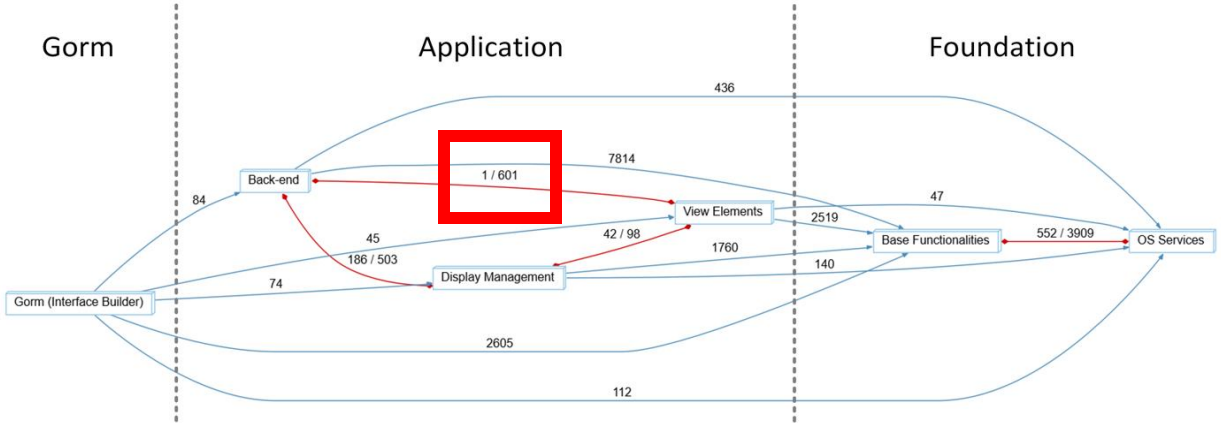
OVERVIEW OF CONCRETE ARCHITECTURE



DIVERGENCES

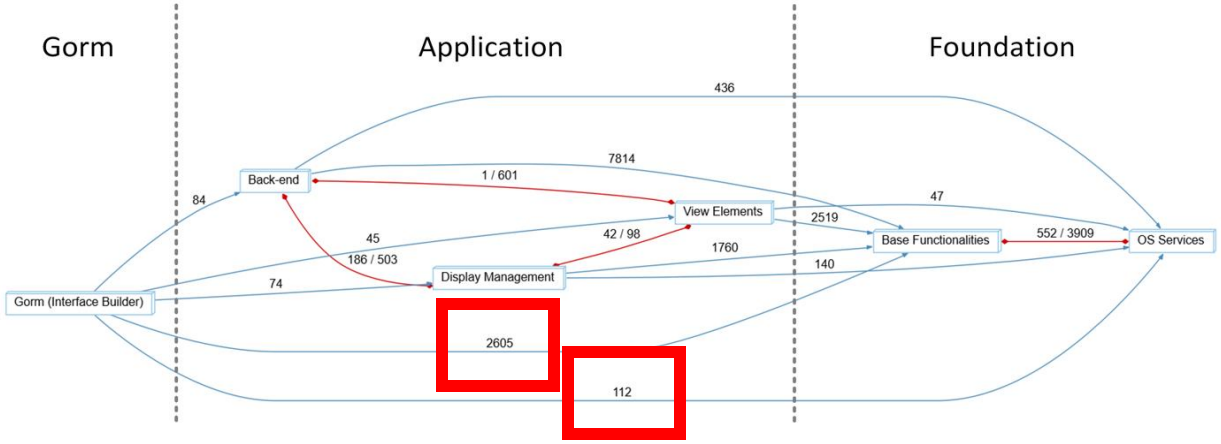
REFLEXION ANALYSIS: HIGH-LEVEL ARCHITECTURE

- **Divergence 1:** two-way dependency between *Back-end* and *View Elements*
  - **Rationale:** *NSStoryboardSegue* invokes the *setViewController* methods provided by *View Elements* to switch control between different views



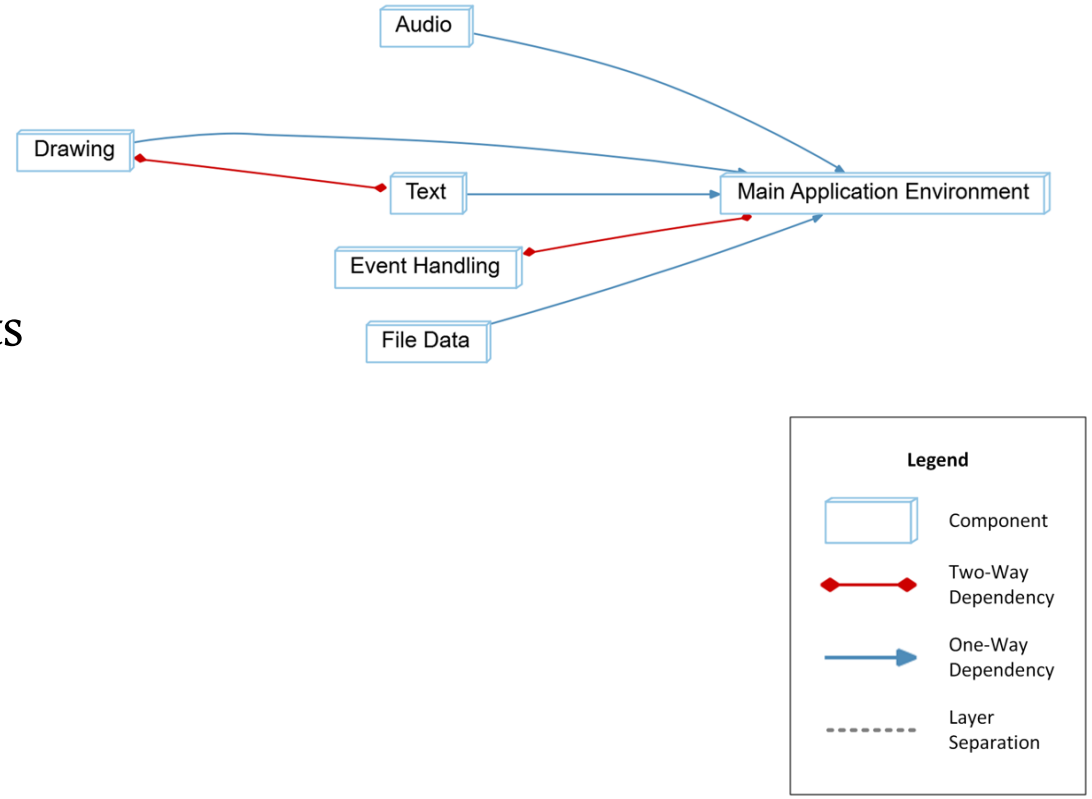
REFLEXION ANALYSIS: HIGH-LEVEL ARCHITECTURE

- **Divergence 2 + 3:** one-way dependency between *Gorm* and *Foundation* layer
  - **Rationale:** Additional functionalities implemented in the *Gorm* layer that directly reference the *Foundation* components, rather than propagating requests through the *Application* layer (e.g., XLIFF support)

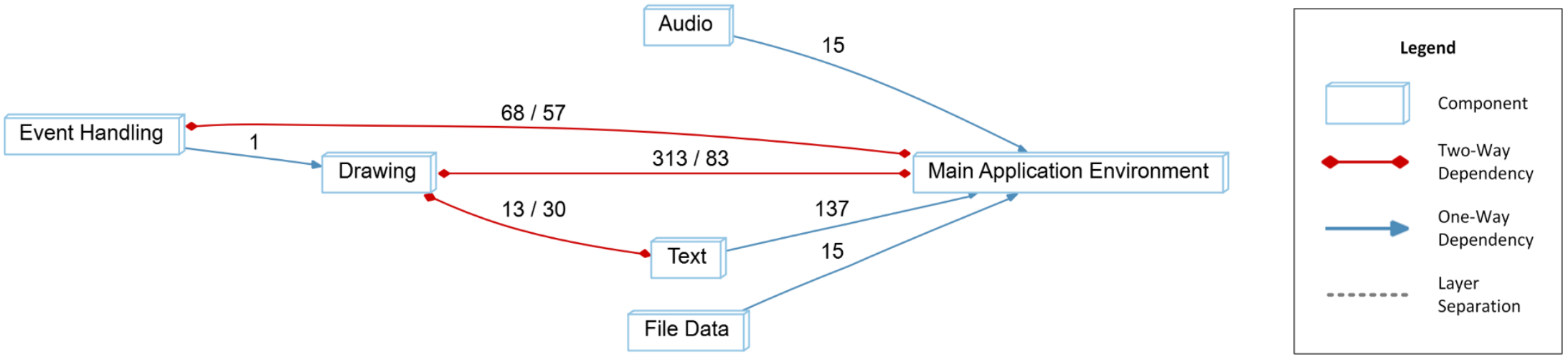


CONCEPTUAL ARCHITECTURE: BACK-END

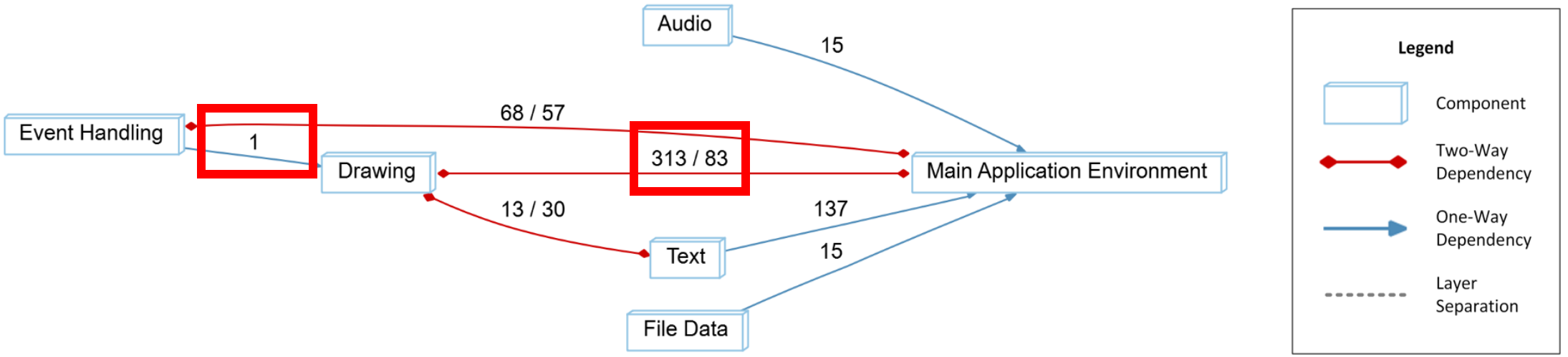
- **Main Application Environment:** provides the application's runtime environment
- **Event Handling:** manages user input events
- **Drawing:** manages display of graphical elements
- **Text:** manages display of textual elements
- **File Data:** manages file data in application
- **Audio:** manages audio input/output



CONCRETE ARCHITECTURE: BACK-END



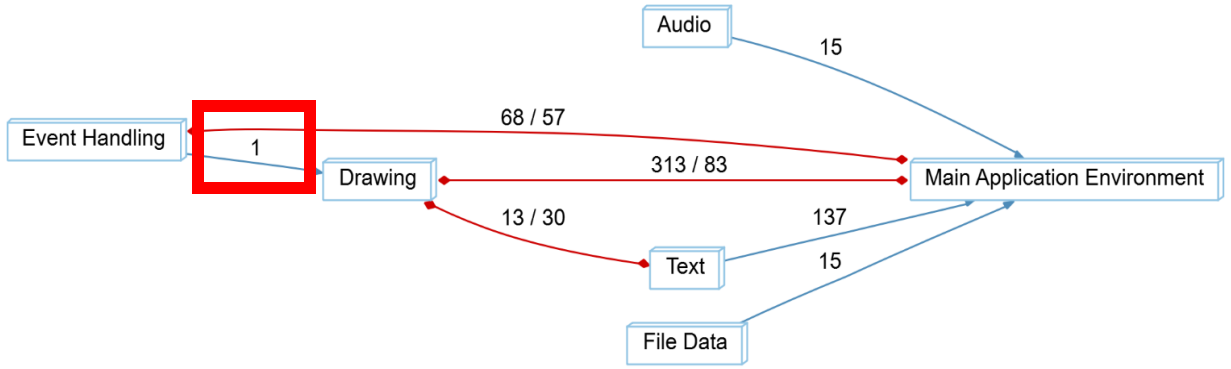
CONCRETE ARCHITECTURE: BACK-END



DIVERGENCES

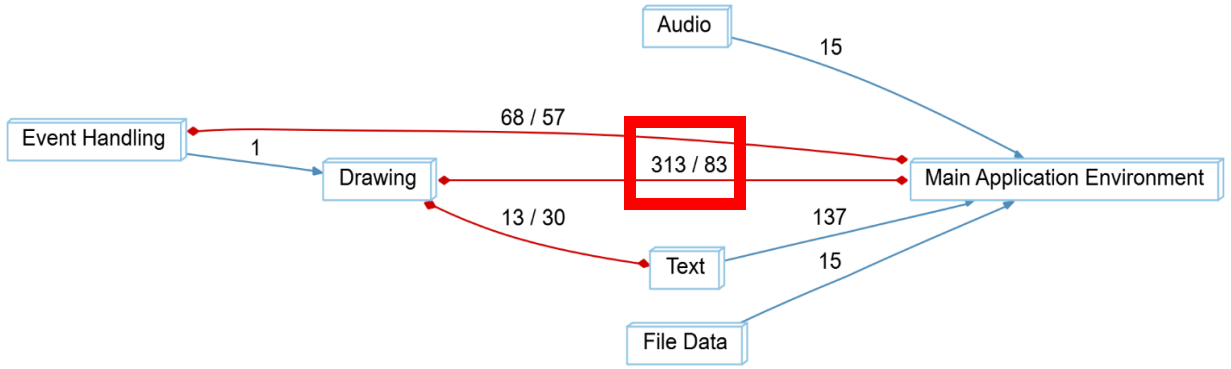
REFLEXION ANALYSIS: BACK-END

- **Divergence 1:** one-way dependency between *Event Handling* and *Drawing*
  - **Rationale:** False positive; *NSControl* checks *type* in *NSEvent* (*Event Handling*), but Understand believes it is checking *type* in *NSColor* (*Drawing*)



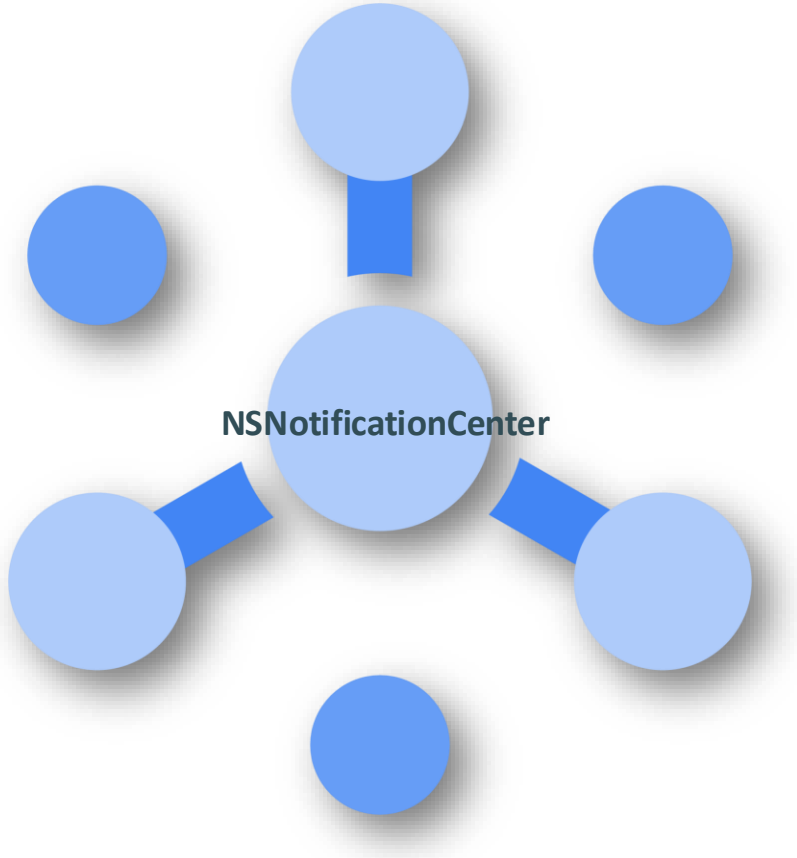
REFLEXION ANALYSIS: BACK-END

- **Divergence 2:** two-way dependency between *Main Application Environment* and *Drawing*
  - **Rationale:** *Main Application Environment* needs graphics contexts from *Drawing* (e.g., *ARTContext*) to know which how to communicate with the display server



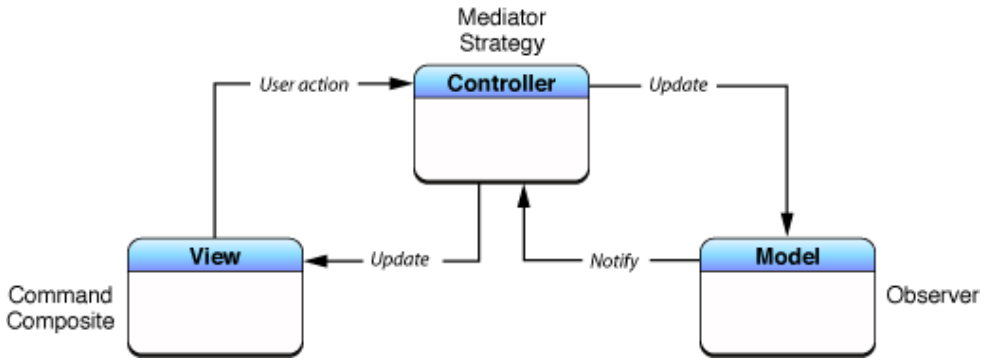
ALTERNATIVE STYLES

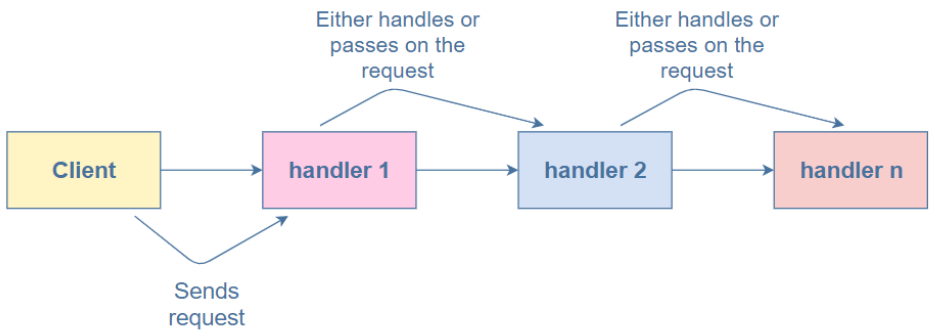
- **Publish-subscribe style:**
  - Implemented via ***NSNotificationCenter***
  - Objects subscribe to *NSNotificationCenter* and respond to events
  - Not chosen due to not representing the overall main architecture



DESIGN PATTERNS

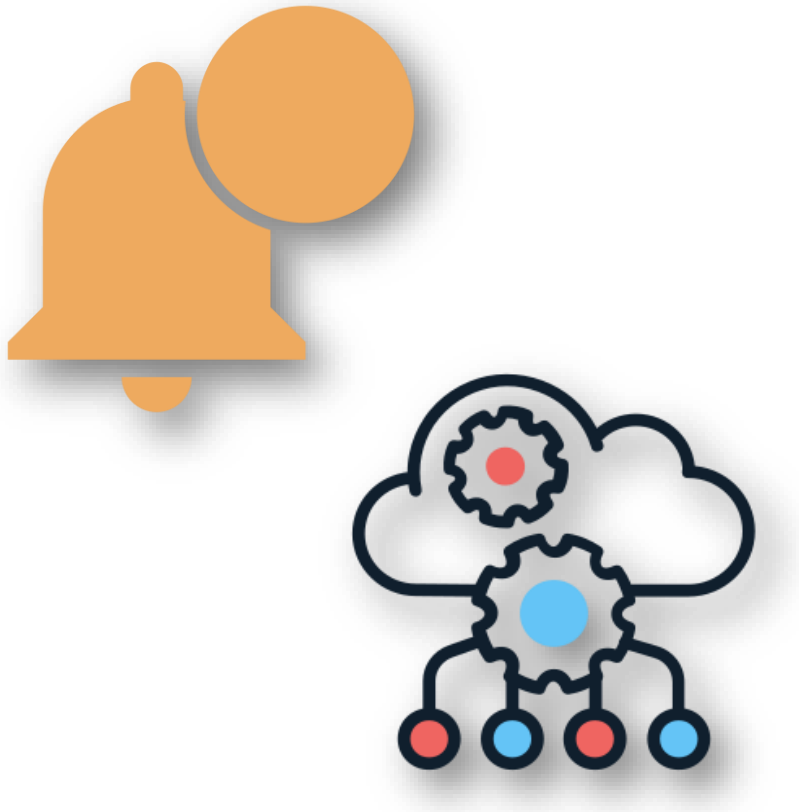
- **Model-View-Controller:** managed via *NSView* and *NSController* classes
- **Chain of Responsibility:** implemented via *NSResponder* classes, which provide the backbone for the responder chain





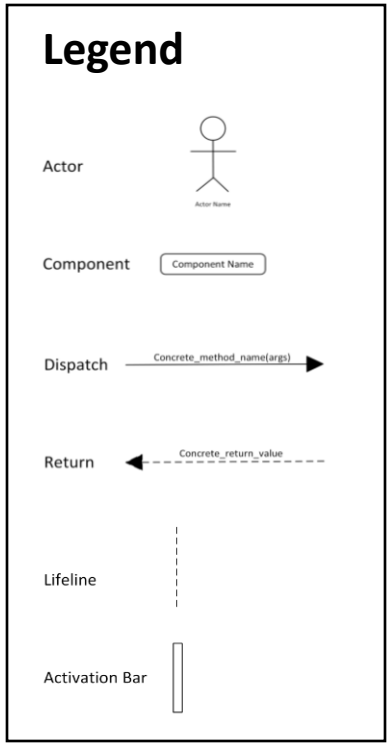
CONCURRENCY AND SYNCHRONIZATION

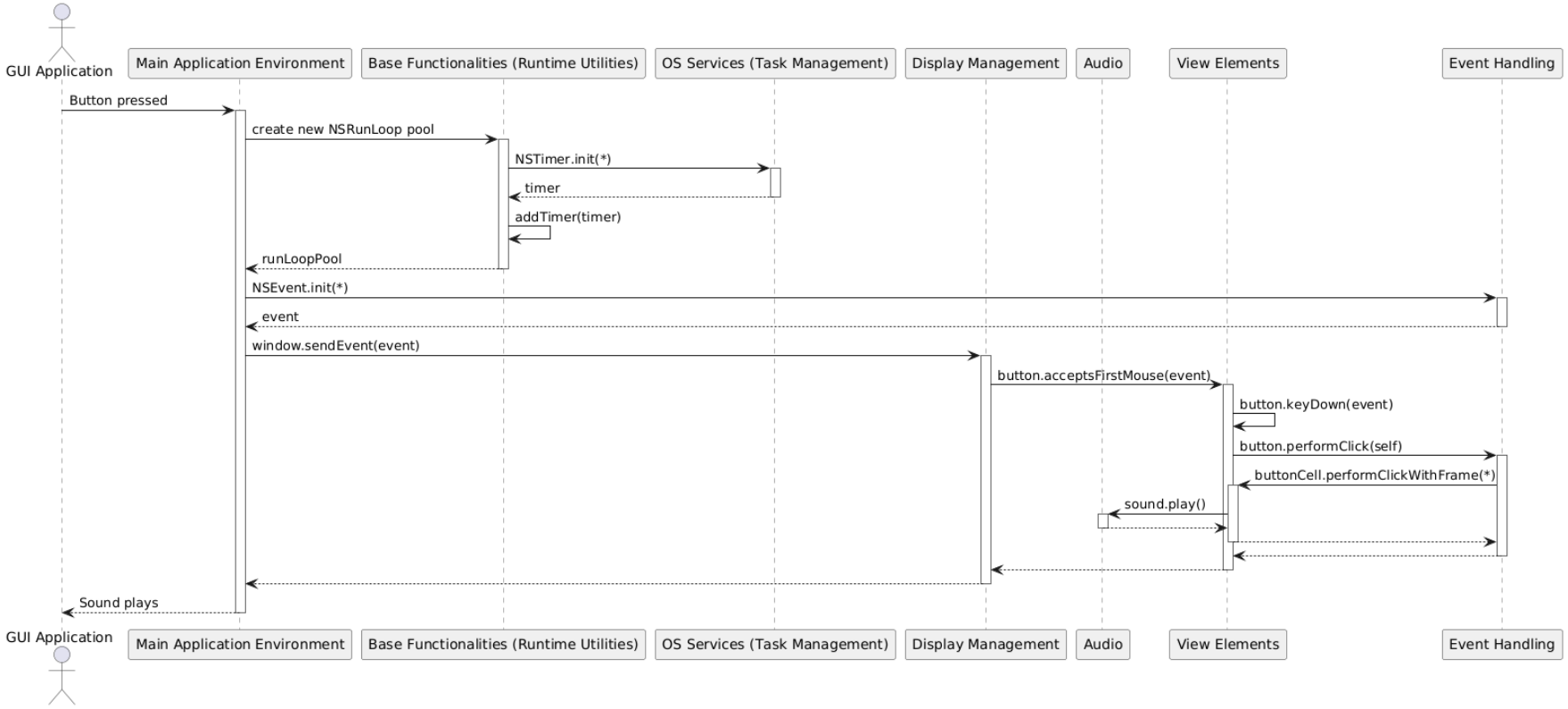
- **Base Functionalities:**
  - ***NSNotificationCenter*:** synchronous processing of object notifications
  - ***NSNotificationQueue*:** asynchronous processing of object notifications
  - ***NSDistributedNotificationCenter*:** processing of object notifications between different processes and/or hosts
- **OS Services:**
  - ***Task Management*:** managing IPC, concurrent resource allocation, multithreading



USE CASE:

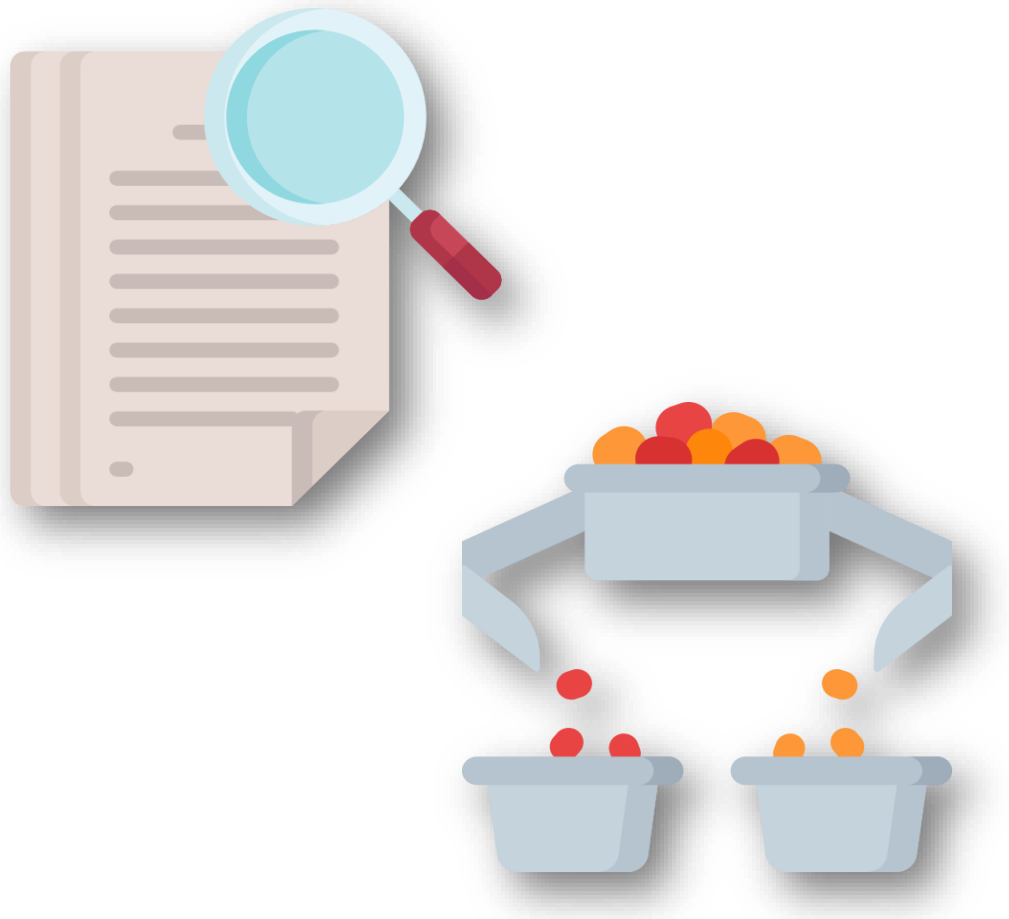
A user presses a button in a GNUstep application, which plays a sound





LIMITATIONS

- Difficult to identify **main functionalities** of classes due to **limited documentation** in headers
- Classes with **multiple different functionalities** can only be classified to **one component**



CONCLUSIONS & LESSONS LEARNED

- **Key features:** layered + object-oriented architecture, MVC/CoR design patterns
- **Key Divergences:**
  - Back-end ↔ View Elements
  - Gorm → Foundation
  - Drawing ↔ Main Application Environment
- Divergences must be **carefully assessed** for **correctness** and **beneficiality**