GNUSTEP ENHANCEMENT PROPOSAL: LLM SUPPORT

Group 12

CISC/CMPE 322/326

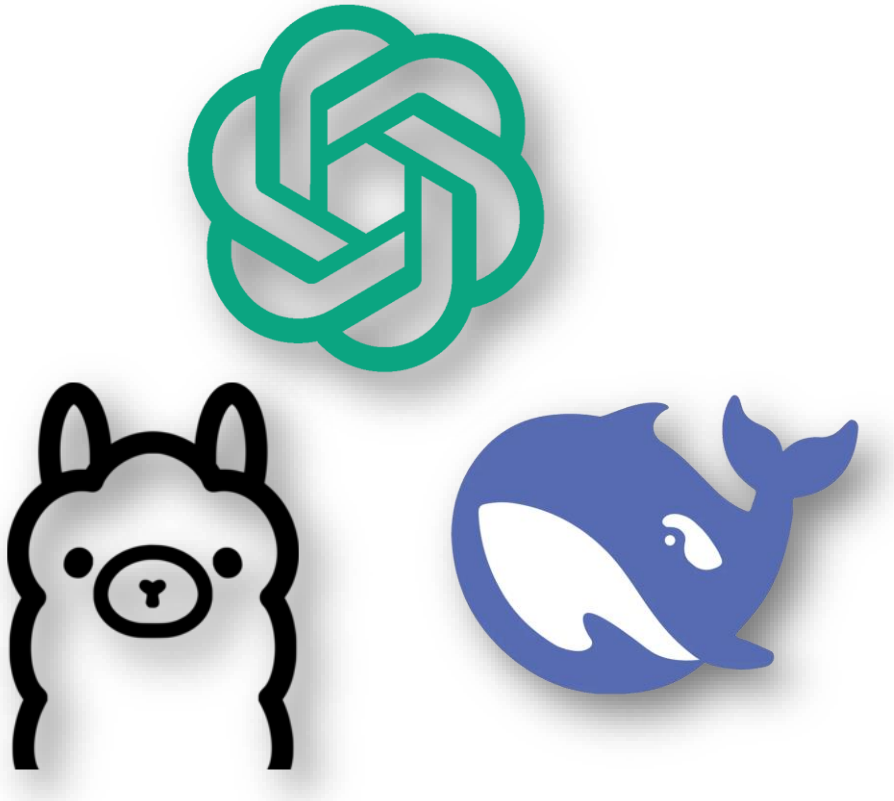
Link to video: https://youtu.be/HsMUmAV14n0

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

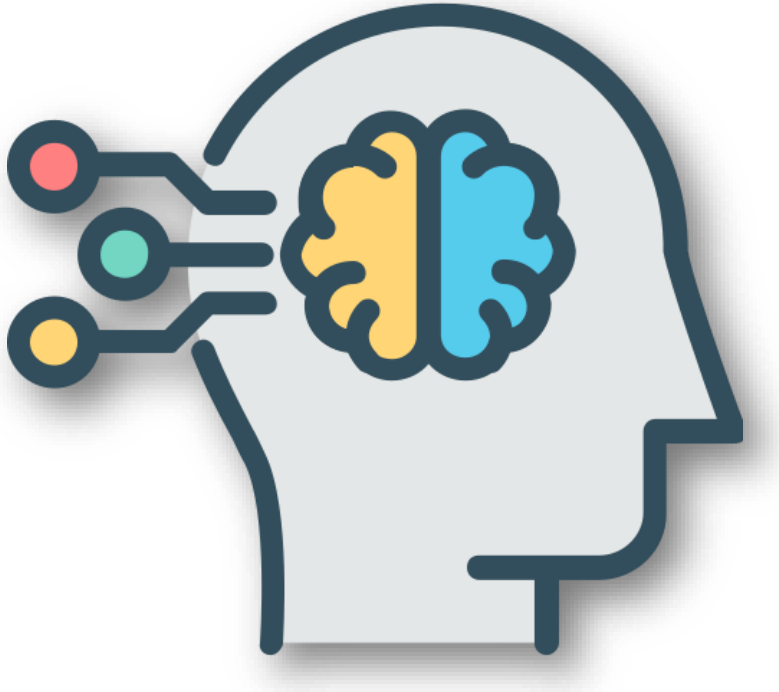
OVERVIEW AND MOTIVATION: LLM SUPPORT

- LLMs are being adopted by various industries at a **rapidly increasing pace**
- GNUstep's framework must stay **up-to-date** with these demands to remain competitive
- **Main focus:** enabling LLM integration into GNUstep applications via simple interfaces that connect to external APIs



NEW COMPONENT: LLM SUPPORT

- Located in **Application layer**
- Handles LLM-related events, calls APIs for external LLMs, and manages input/output/logging of such processes
- Provides GUI elements with **LLM Controllers** to handle LLM events

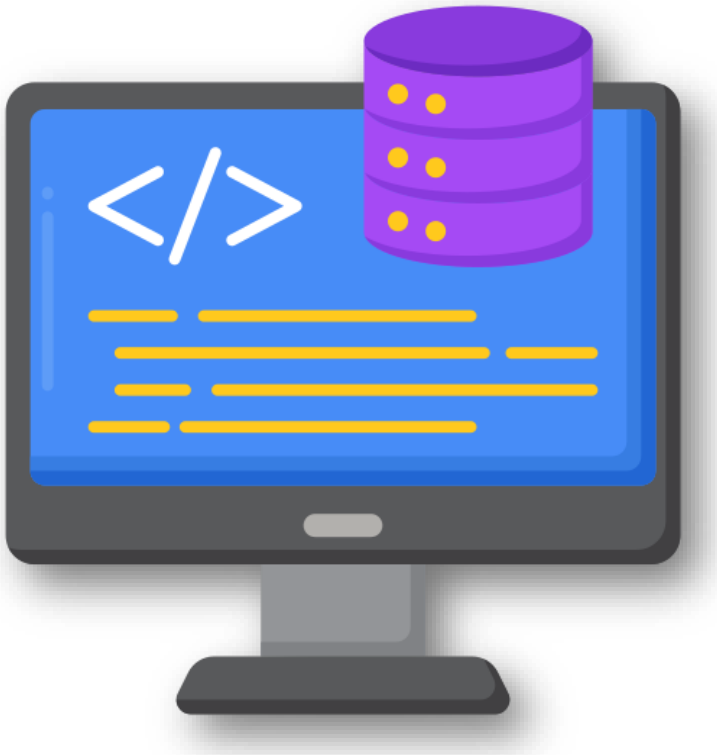


COMPONENT INTERACTIONS: LLM SUPPORT (1)

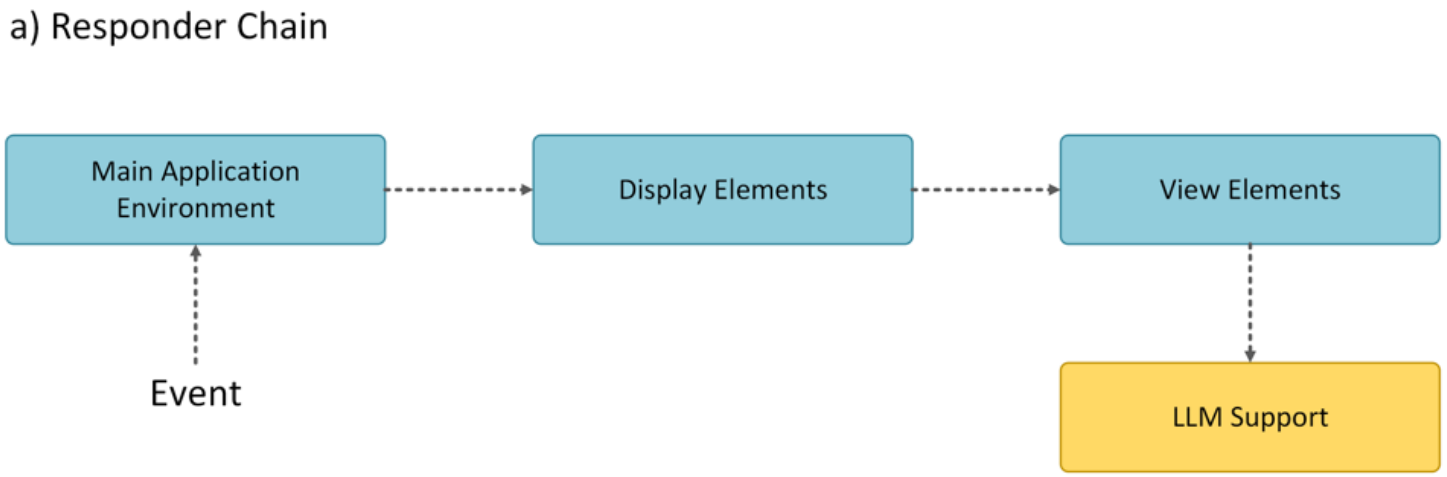
- Foundation Layer dependencies:
  - ***Runtime Utilities*:** error handling, logging, and task management functionalities
  - ***Value Data*:** data structures that help manage inputs/outputs to API calls
  - ***OS Services*:** networking, IPC, and file management functionalities

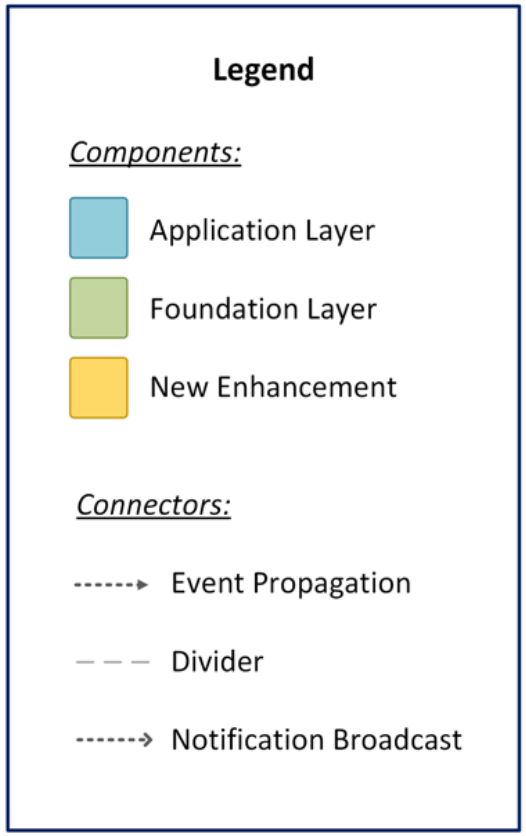
COMPONENT INTERACTIONS: LLM SUPPORT (2)

- Application Layer dependencies:
  - ***Main Application Environment*:** connecting to external LLM interfaces
    - Two-way dependency; *LLM Support* is needed to provide information for initializing and managing LLM Controllers
  - **Other *Back-end* Components:** handling different outputs from API calls



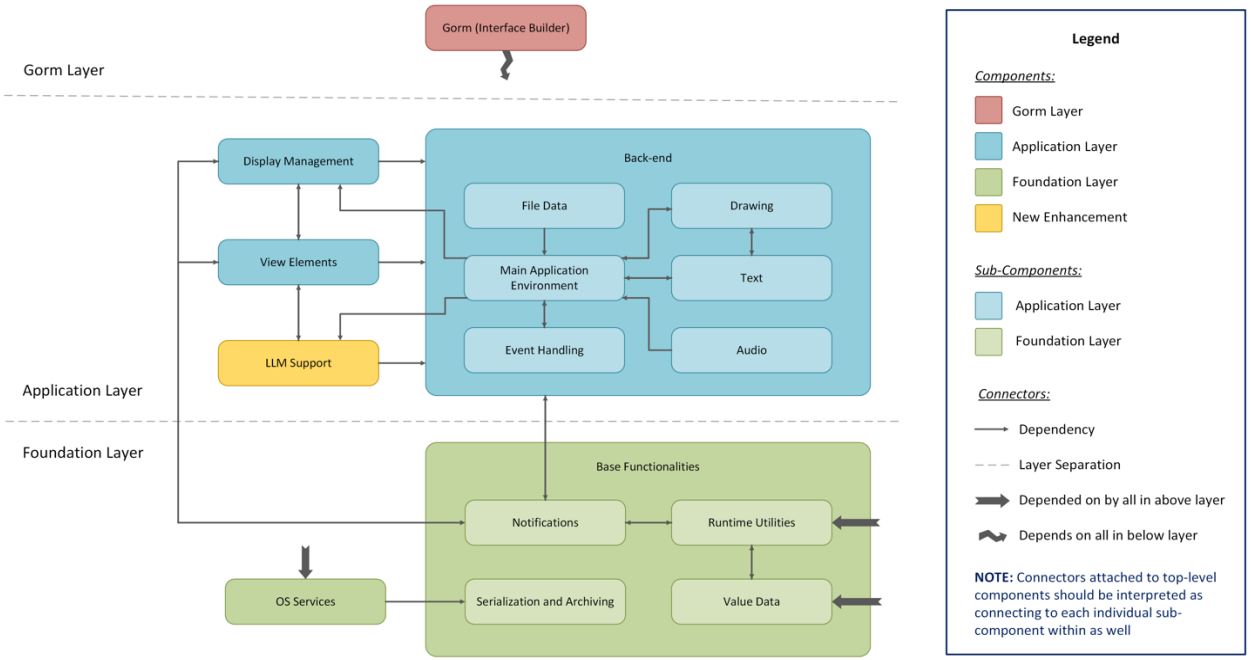
APPROACH 1: RESPONDER CHAIN INTEGRATION





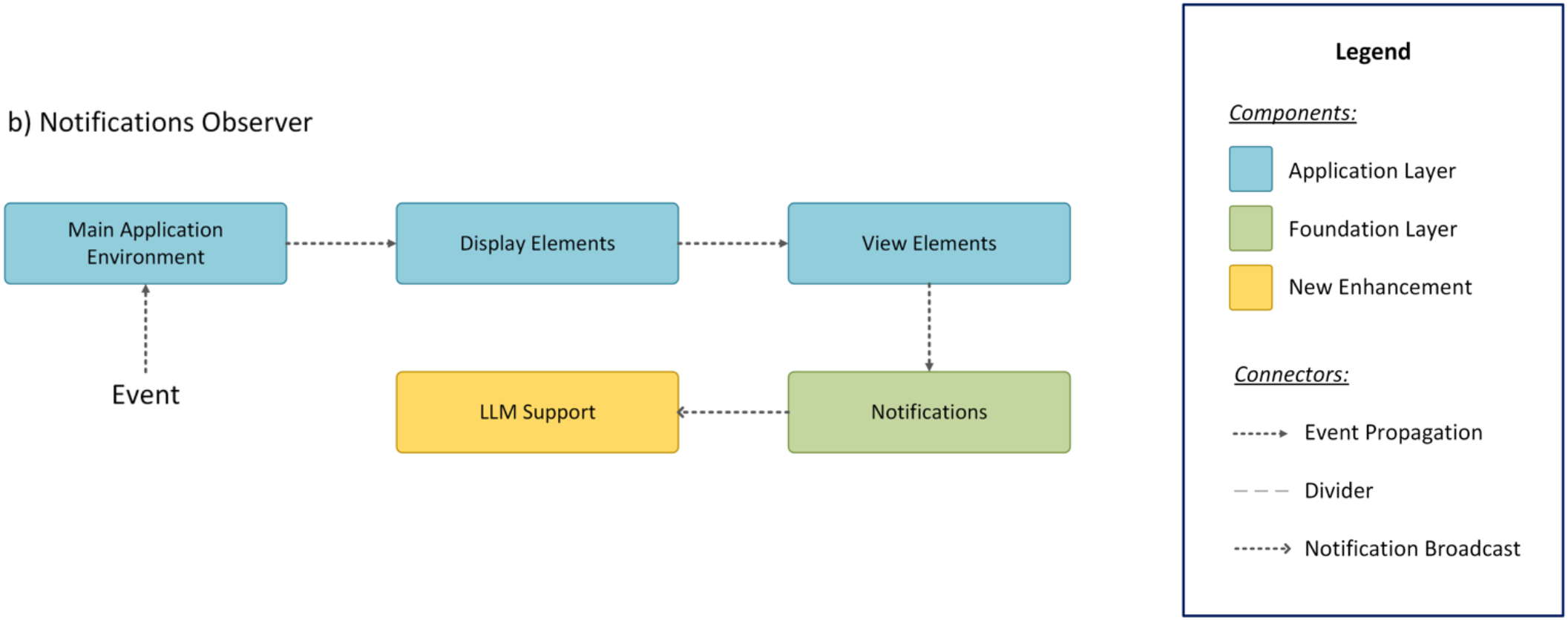
APPROACH 1: IMPACTED SUBSYSTEMS

- New two-way dependency between ***LLM Support*** and ***View Elements*** to facilitate event propagation
- Adjustments to ***libs-gui*** library:
  - New files for **LLM Controller** classes
  - Modifications to **Storyboard** classes in ***Main Application Environment*** to accommodate LLM Controllers
  - Classes in ***Event Handling*** should account for LLM Controllers
  - Modification to event dispatch methods in ***View Elements***



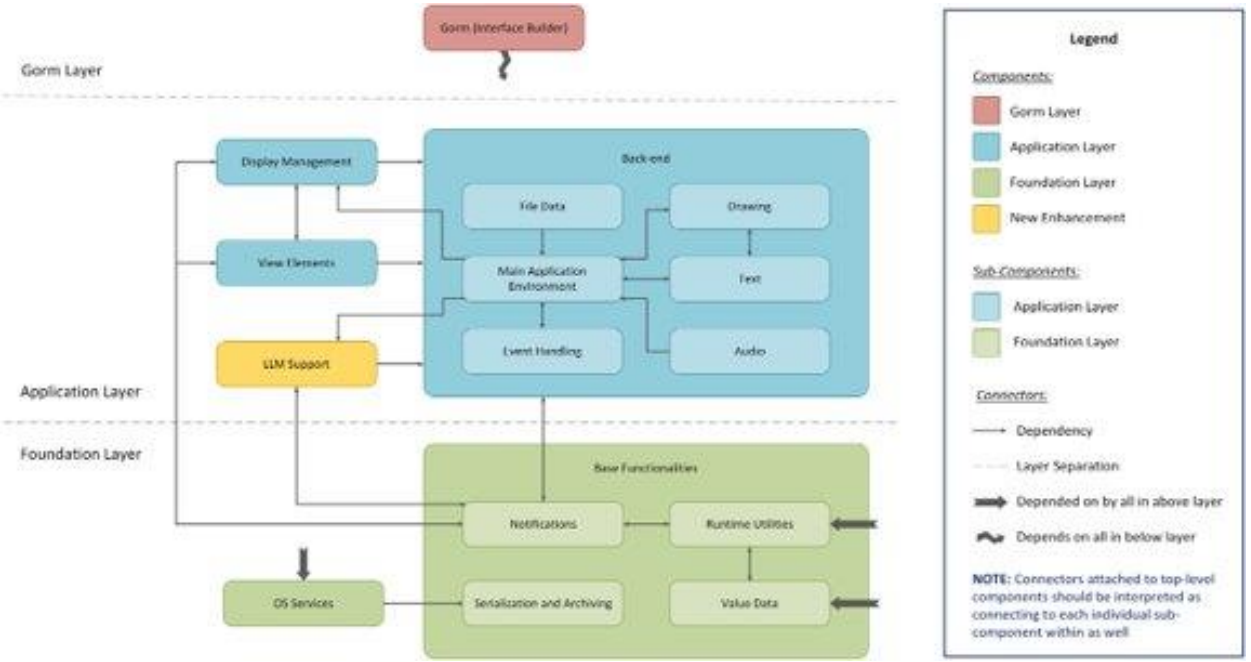
Architectural Style:
Layered, Object-Oriented

APPROACH 2: NOTIFICATION OBSERVERS



APPROACH 2: IMPACTED SUBSYSTEMS

- New two-way dependency between ***LLM Support*** and ***Notifications*** to facilitate pub-sub communication
- Adjustments to ***libs-gui*** library:
  - Similar to Approach 1
- Adjustments to ***libs-base*** library
  - Modifications to classes in ***Notifications*** to recognize LLM Controllers as observers
  - New files for **new types of notifications** that handle data and operations related to LLMs



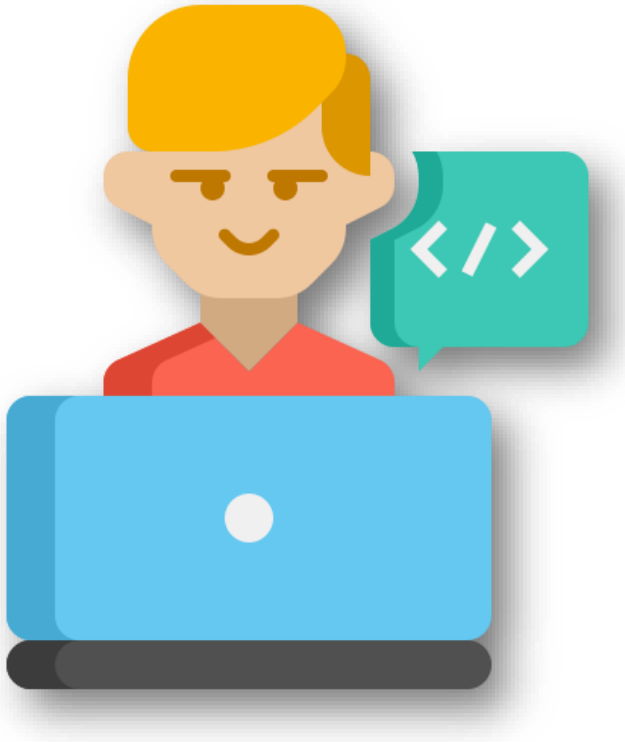
Architectural Style:
Layered, Object-Oriented + Pub-Sub

SEI SAAM ANALYSIS: STAKEHOLDERS (1)

- **GNUstep Contributors:**
  - Responsible for maintaining core GNUstep framework
  - Must maintain, debug, and evolve *LLM Support* component
- **Major Considerations and NFRs:**
  - **Maintainability:** must follow established GNUstep design principles
  - **Evolvability:** should be able to easily integrate new LLM APIs

SEI SAAM ANALYSIS: STAKEHOLDERS (2)

- **GNUstep Developers using Gorm:**
  - Will integrate functionalities provided by *LLM Support* into their GUI applications
- **Major Considerations and NFRs:**
  - **Scalability:** apps should remain operational under heavy request load
  - **Ease-of-use:** new features should be easy for developers to use in their apps



SEI SAAM ANALYSIS: STAKEHOLDERS (3)

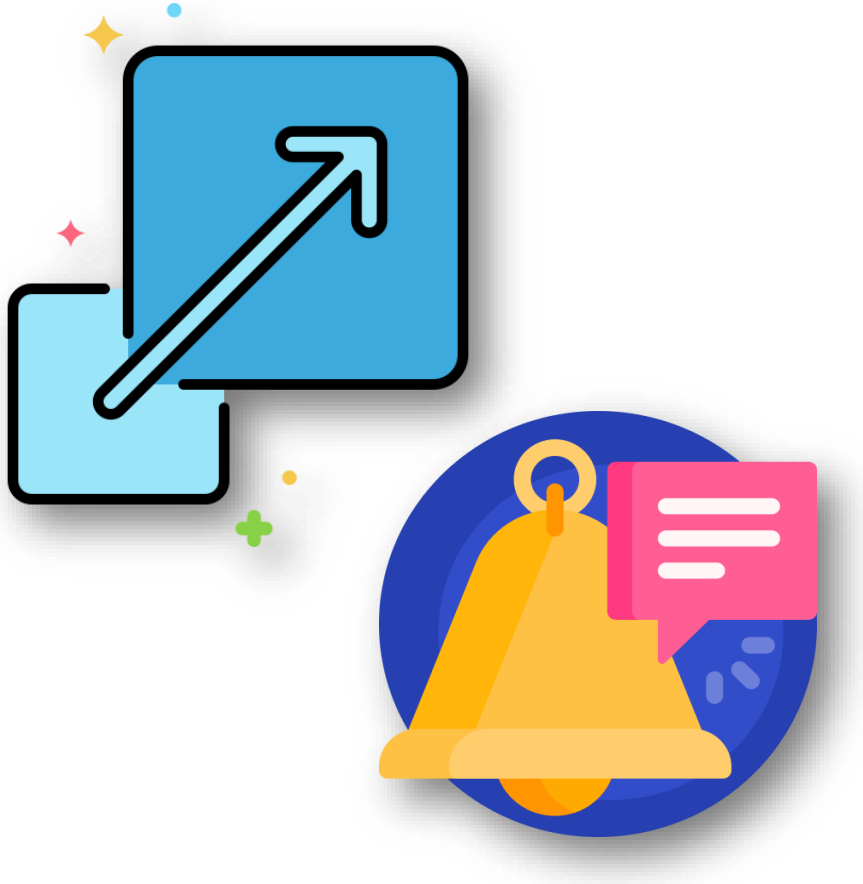
- **End Users of GNUstep Applications:**
  - Will interact with features in applications enhanced by *LLM Support* (e.g., text-to-speech)
- **Major Considerations and NFRs:**
  - **Performance:** should provide fast and accurate responses
  - **Security:** should protect sensitive data

SEI SAAM ANALYSIS: APPROACH COMPARISON

NFR	Approach 1	Approach 2
Maintainability	Pre-existing implementation of responder chain allows for easy integration into the system	Wide range of observers makes it difficult to trace notifications in the system for debugging
Evolvability	Tight coupling between *LLM Support* and *View Elements* forces them to evolve alongside each other	*LLM Support* receives events via *Notifications*, allowing it to operate independently from *View Elements*
Performance	Responder chain provides more consistent response times, but can suffer from high request load	Notification system has unpredictable response times, but asynchronous operations allow for high scalability
Security	Low scalability may make system more prone to DDoS attacks	Malicious observers could potentially connect to notification system, resulting in data leak

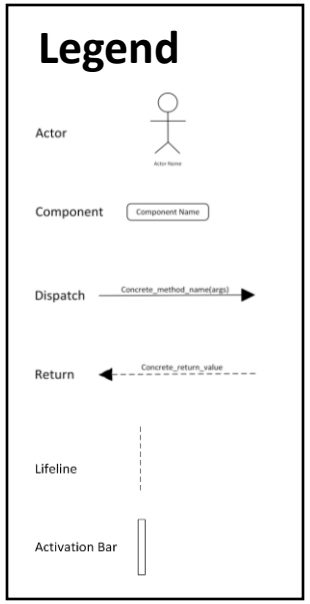
SEI SAAM ANALYSIS: CHOSEN APPROACH

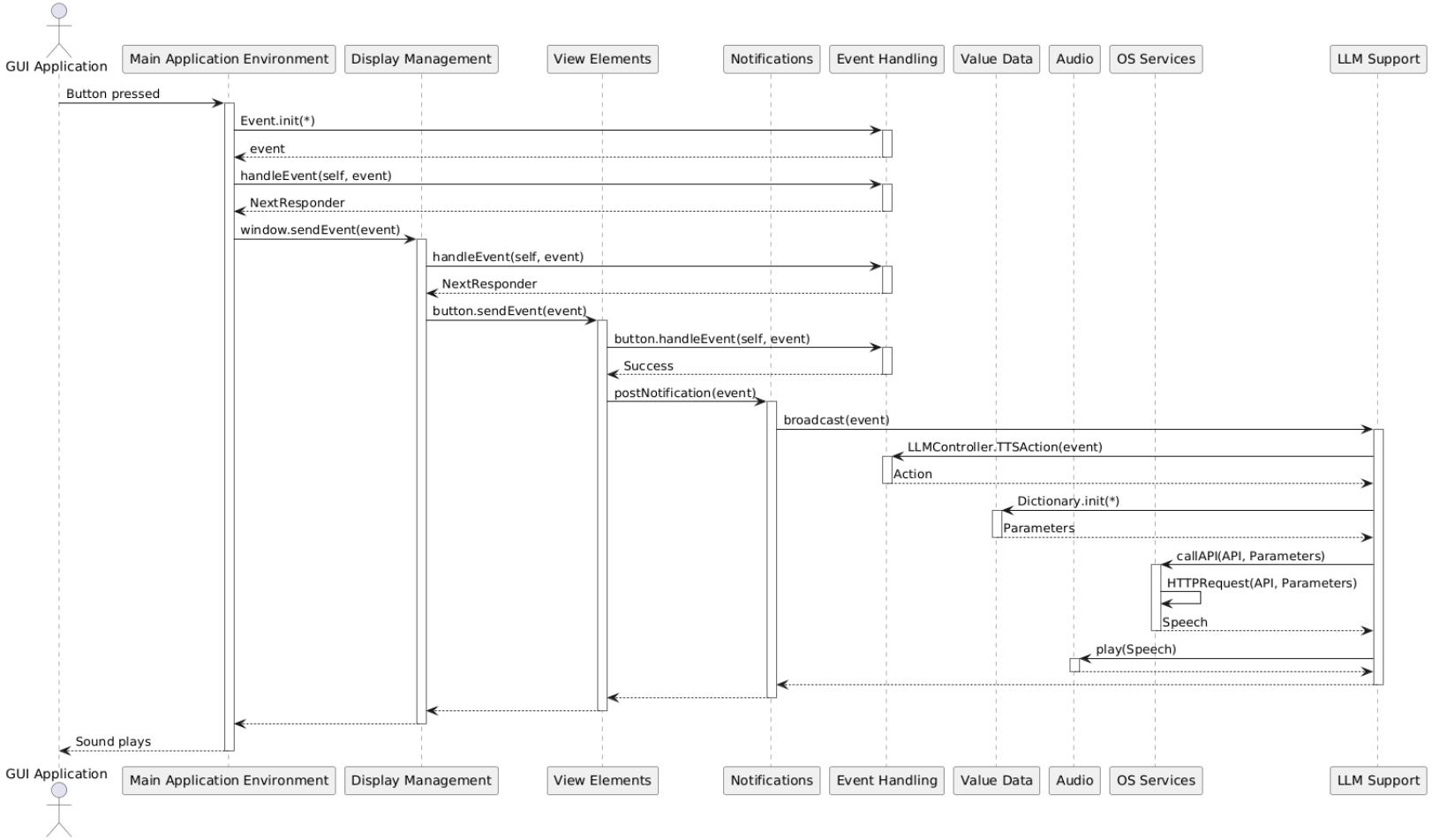
- **Approach 2** was chosen due to having more significant benefits and less detrimental shortcomings
  - Much **cheaper to evolve**, which is important in the rapidly-evolving field of LLMs
  - Prioritizes **scalability** over **response time consistency**, ensuring the system is always operational
  - Maintainability can be improved by implementing **notification logs**
  - Security can be improved with **encryption** and **firewalls**



USE CASE:

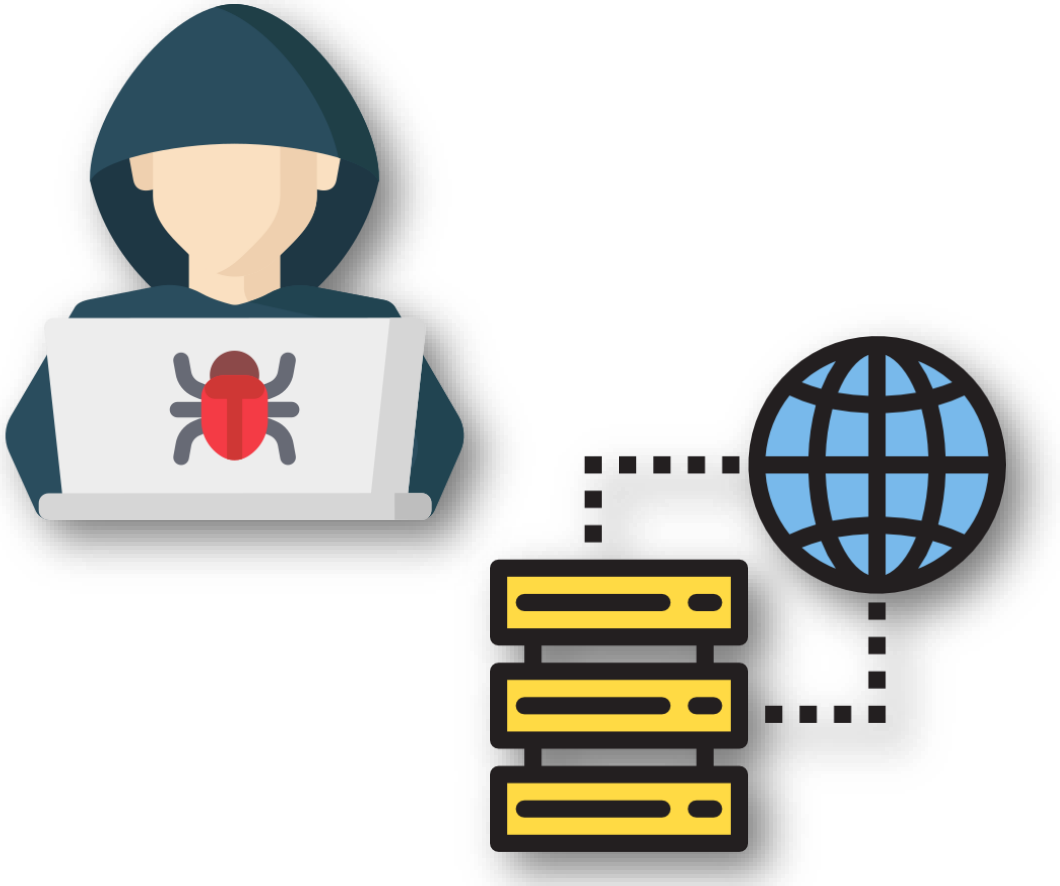
A user presses a button in a GNUstep application, which plays a text-to-speech narration of some text, provided by an LLM





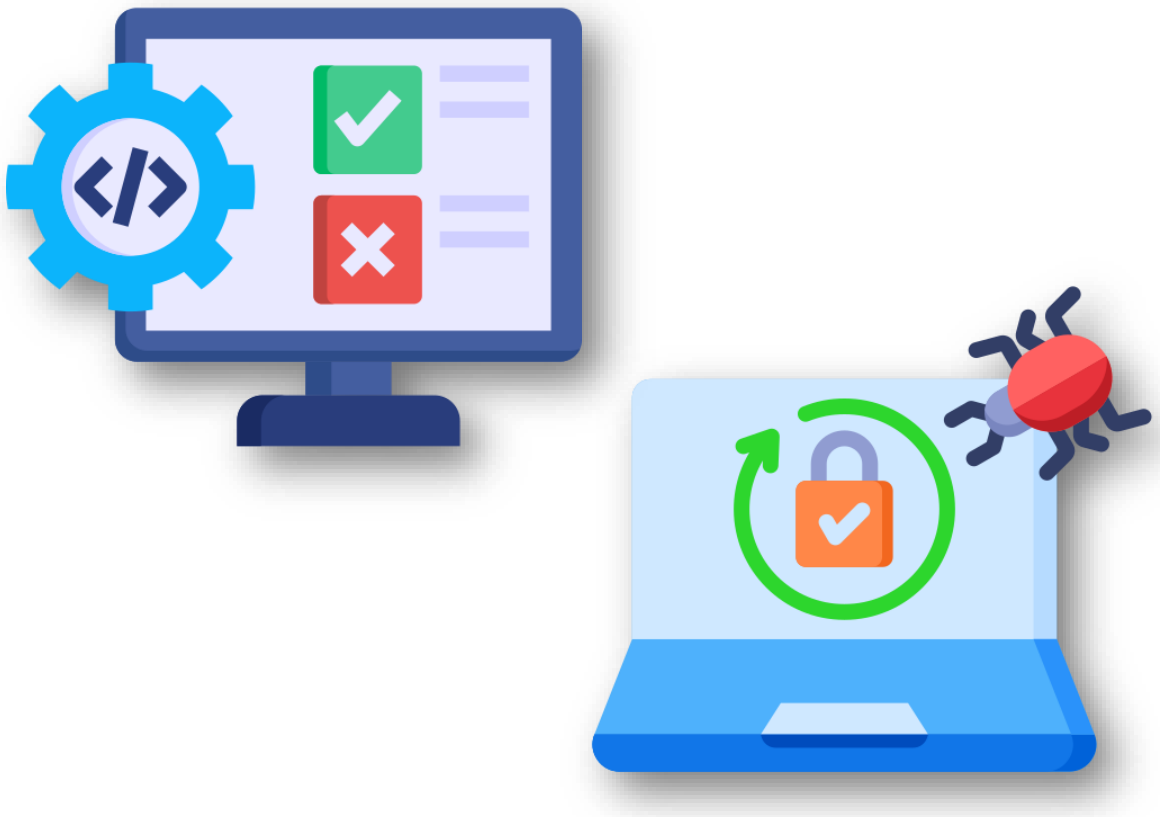
POTENTIAL RISKS AND LIMITATIONS

- **Security:** transmission of sensitive data between application and external interfaces may be intercepted by malicious parties
- **Maintainability:** must constantly remain up-to-date with changes to external APIs
- **Performance:** highly dependent on performance of LLM servers



TESTING PLANS

- **Integration testing:** validating correctness of interactions between GNUstep and external LLMs
- **Regression testing:** identifying unwanted side effects
- **Security testing:** identifying vulnerabilities in API connections
- **Performance testing:** verifying response time and max request load



CONCLUSIONS & LESSONS LEARNED

- **Important considerations:** easy maintainability and evolvability
- **Main performance metrics:** response time and request load
- **Notification Observers approach** covers these requirements more thoroughly
- **Key lessons:** reuse pre-existing architectural styles and design patterns when implementing new features