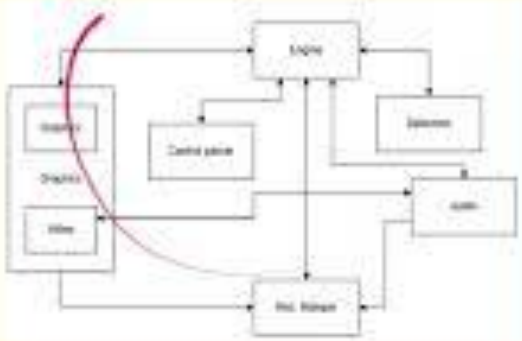
Video Presentation

Conceptual Architecture: SCI

- Contains all essential components to run point and click adventure games.
- Engine is the most crucial, responsible for all functionalities in the game (e.g. camera, collision scripts)
- Control parser is the most simple major component, it acts similar to a keymapper
- Graphics contains two components, Graphics and Video, which handle real time rendering and pre-recorded videos respectively.



A2: Concrete Architecture

Group 9

Xavier Awadalla (Team Leader)
Craig Tylman (Presenter)
Christopher Gil (Presenter)
Sam Tylman
Aaron Rivest
Felix Xing

(group roles listed on next slide)

https://www.scummvm.org https://github.com/scummvm/scummvm

Group member roles

Xavier Awadalla

Concrete architectural derivation process

Aaron Rivest

Sequence diagrams

Christopher Gil

Sequence diagrams, presentation

Craig Tylman

Reflexion analysis, presentation

Felix Xing

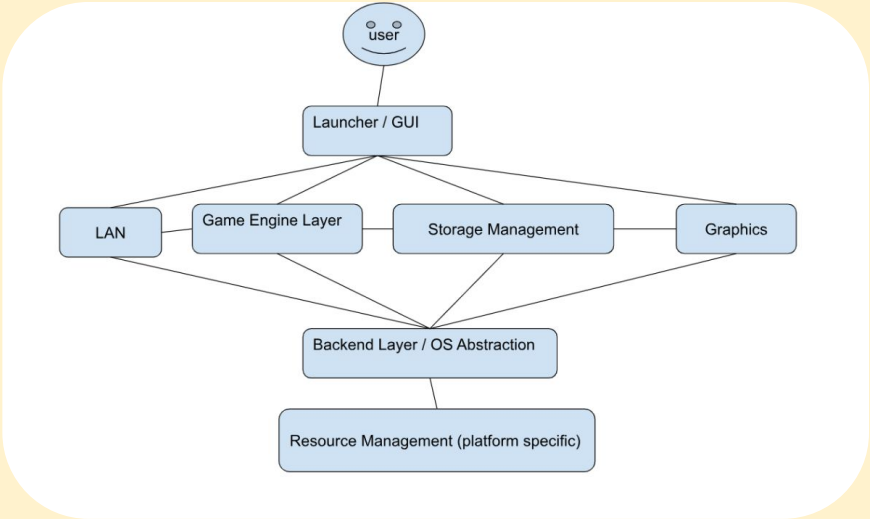
Architecture diagrams and derivation

Sam Tylman

Sci conceptual architecture

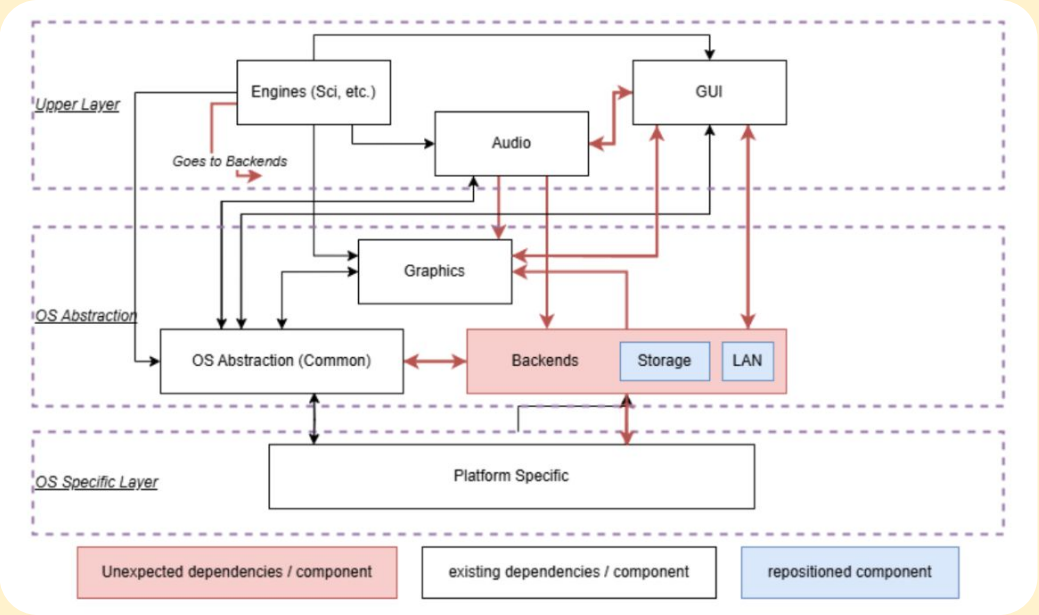
Recall: Conceptual Architecture

- Previously created in A1
- Still holds up, Upper, OS and Platform-specific layers are easy to tell apart.
- No need to update, however, many divergences were found.
- Some component hierarchy changes & unexpected components as well.



Concrete Architecture

- **Layered structure**, with distinct top-level subsystems
- **Alternative:** A Microservices structural style was considered, but discarded as many components were not decoupled, nor functioned independently.
- **Divergences are show in red** (arrows, boxes), and structure/hierarchy changes are blue.
- **Overall structure** remains similar to the conceptual architecture.
- Most components **rely heavily** on the OS Abstraction (Common)



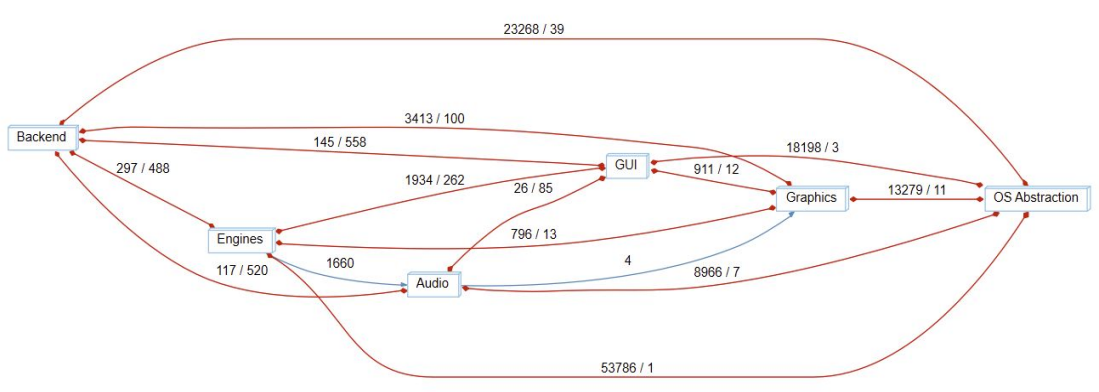
Reflexion Analysis

Several discrepancies from the original conceptual architecture, done in good faith in order to improve the software's performance & streamline development

- **Audio ⇔ GUI**
  - Audio uses GUI to render error messages & play error sounds
  - GUI widgets link to Audio component to change its settings
- **Graphics → GUI**
  - Vector renderer calls GUI to customize shading, overlays, text alignment, among other details.
- **Backend → GUI**
  - Backend uses the GUI modal and dialogue to display messages/error/warnings to the client.
- **Audio → Graphics**
  - Unused import of font, fontman, surface, pixelformat for the Roland MT-32.
- **New Backend Component**
  - Now behaves as a central component. Composed of multiple unique "Backend" core functions and utilities, such as Networking, Cloud Save, TTS, MIDI, and more.

Derivation Process Of Concrete Architectures

- Derived using the Understand tool.
- Creating a new architecture in the Architecture Browser, we could add ‘nodes’ that align with each subcomponent derived in our conceptual architecture.
- Each node had subcomponents added as folders, and an accurate graph for the concrete architecture was created.
- As a group, we had to be smart with how we split the work - proper reflexion analysis required the creation of this graph ahead of time



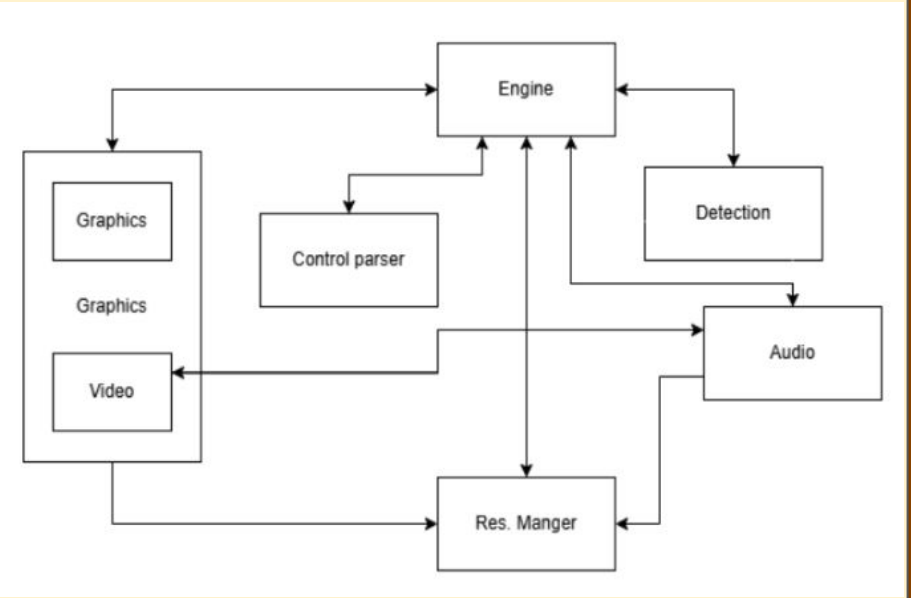
Derivation Process (cont.)

- Initial conceptual arch. and new concrete arch. were compared: convergences and divergences in subcomponents and interactions were identified.
- Git blame was used to create sticky notes & find out rationale for some divergences.

An example was the Audio→Graphics dependency added decades ago, with seemingly no use. It served a purpose back then, but has since been phased out entirely.

Conceptual Architecture: SCI

- Contains all essential components to run point and click adventure games.
- Engine is the most crucial, responsible for all functionalities in the game (e.g. camera, collision scripts)
- Control parser is the most simple major component, it acts similar to a keymapper.
- Graphics contains two components, Graphics and Video, which handle real time rendering and pre-recorded videos respectively.



Reflexion Analysis (SCI)

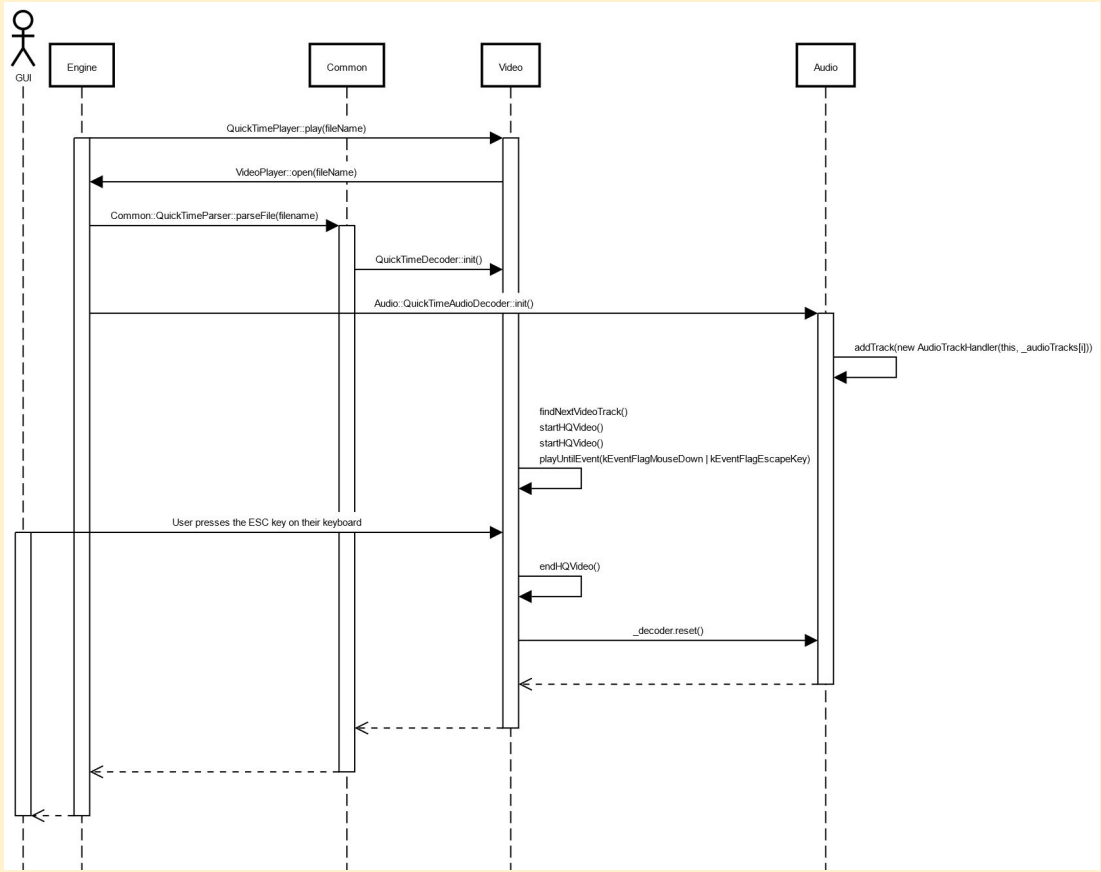
The Concrete architecture of SCI contains all of the essential components for the engine to run point and click adventure games, with some divergences from the conceptual.

Many of SCI's divergences appear to relate to its "vocabulary" system, which functions as a dictionary.

- **Graphics → Control Parser**
  - Graphics & the SCI menu call parser for vocabulary lookups and actions.
- **Video → Resource Manager**
  - "Robot video" decoder uses the class "DecompressorLZS" from resource manager.
- **Graphics → Audio**
  - Video player connects to the audio mixer to decode sound
  - GfxMenu methods contain an optional argument to pause all sound playback
- **Control Parser ⇔ Resource Manager**
  - Frequent calls from Control Parser to Resource Manager for reference and lookup vocabulary data
  - Resource Manager has a single call to Control Parser when initializing SCI games. It accesses a single constant: VOCAB_RESOURCE_SELECTORS=997

Sequence Diagram: Running an introductory cutscene in ScummVM

Concurrency is simple. It may look like many actions are happening simultaneously, however, they are not async calls as all components have to wait for a response to determine how to proceed (or exit if an error occurs)



Conclusions & Lessons Learned

- More divergences than expected! But why…?
- Many internal changes over the years, with erosion.
- Improvements can be made!
  - Component decoupling
  - Code cleanup, maintenance & documentation
- Not many limitations during the derivation process. Understand helped greatly in visualizing dependencies and file structures.
- The source code also helped connect missing links and rationale for divergences by directly inspecting what the function calls did.
- Of course, we are not as experienced with this software as the actual developers and other stakeholders.

Thank you!

Concrete vs Conceptual architecture of scummvm