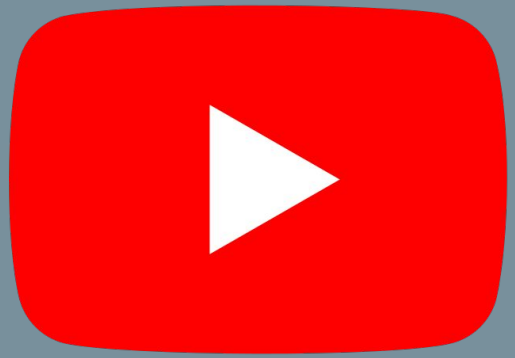


YouTube Link

…and yes, with the right add-on, you can watch in in Kodi.

Copy this link:

https://youtu.be/2UjSwFEKjqg

or just click here!

In Medias Res:
Kodi's Concrete Architecture

Group 2 - ArchAngels

Presentation Scripter, Report Writer- Bianca Bucchino: 20blb@queensu.ca
Architectural Analyst, Report Writer - Adam Clarke: 21asc6@queensu.ca
Presentation Designer, Narrator, Report Editor - Christian Higham: 20csdh@queensu.ca
Architectural Analyst, Report Writer - Omar Ibrahim: 20omha@queensu.ca
Architectural Analyst, Reflexion Analyst, Report Writer - Owen Rocchi: 19omr6@queensu.ca
Group Leader, Architectural Analyst, Report Writer, Diagram Designer - Aidan Sibley: 20ajs18@queensu.ca

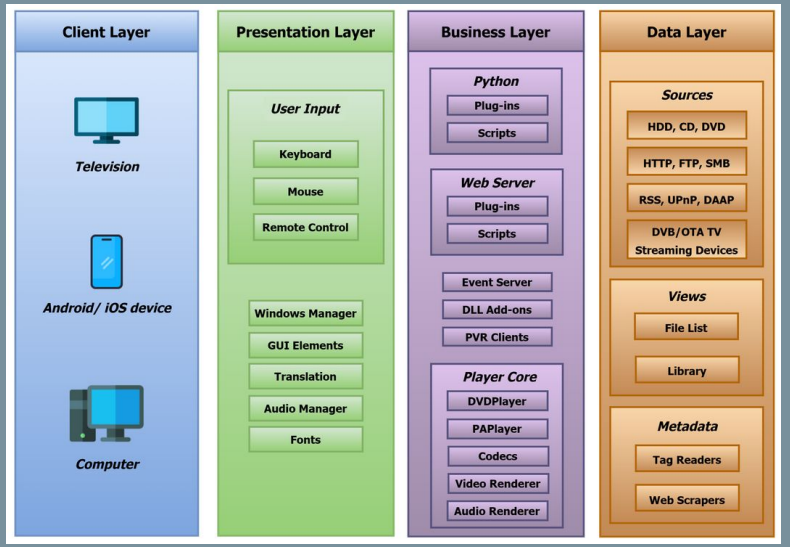
Presentation Overview

- What is Concrete Architecture?
- Kodi's Concrete Architecture
  - Architectural Style
  - Alternative points of view
  - Derivation Process
- Subsystem Architecture
- Conceptual Vs. Concrete
- Reflexion Analysis
  - Concrete Architecture
  - Subsystem Architecture
- Lessons Learned
- Conclusion

Concrete Architecture

- Concrete Architecture is the implemented structure of a software's code
- It also defines how different components within software interact with one another

Kodi's Concrete Architecture, as shown on the official wiki



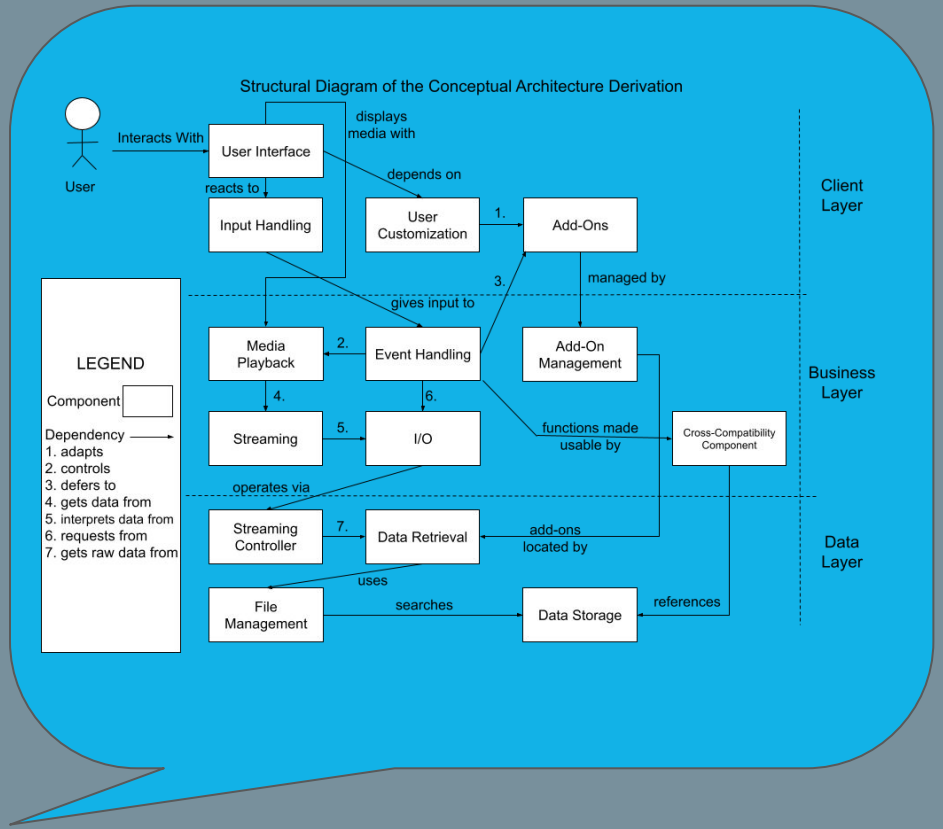
https://kodi.wiki/view/Architecture

What Is Kodi's Concrete Architecture?

What it Isn't

- Initially, we thought Kodi's architecture would have been layered, in line with our conceptual architecture and Kodi's documentation
- However, it appears to take on more of a repository style due to increased

Our Derivation of Kodi's Conceptual Architecture

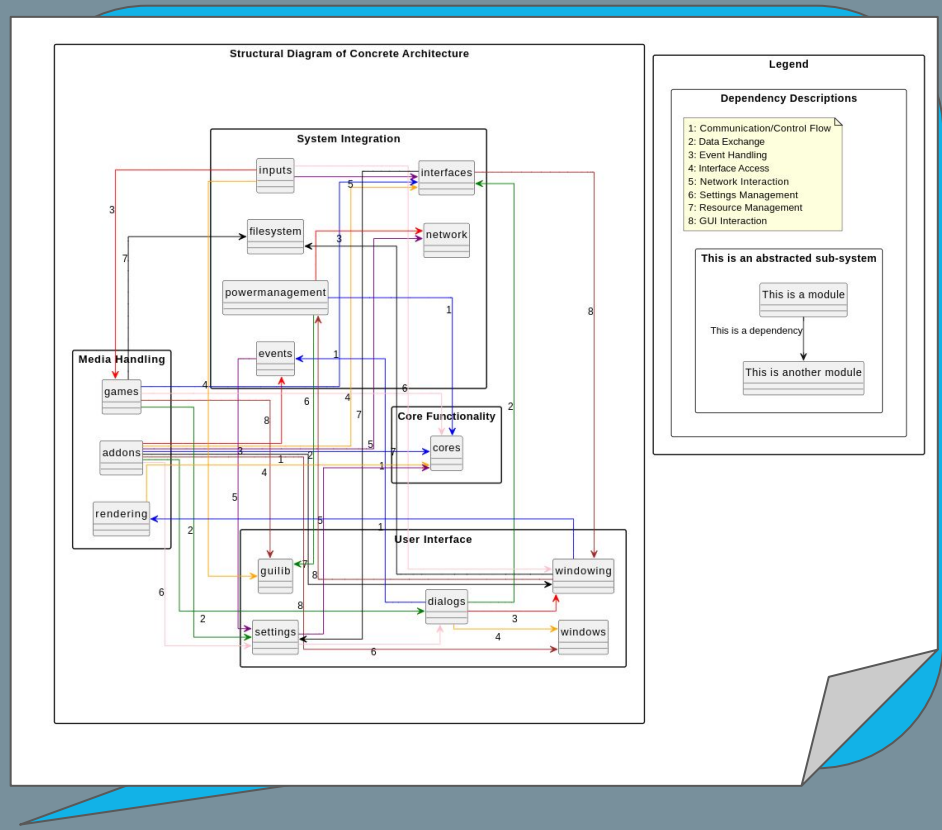


What Is Kodi's Concrete Architecture?

What it Is

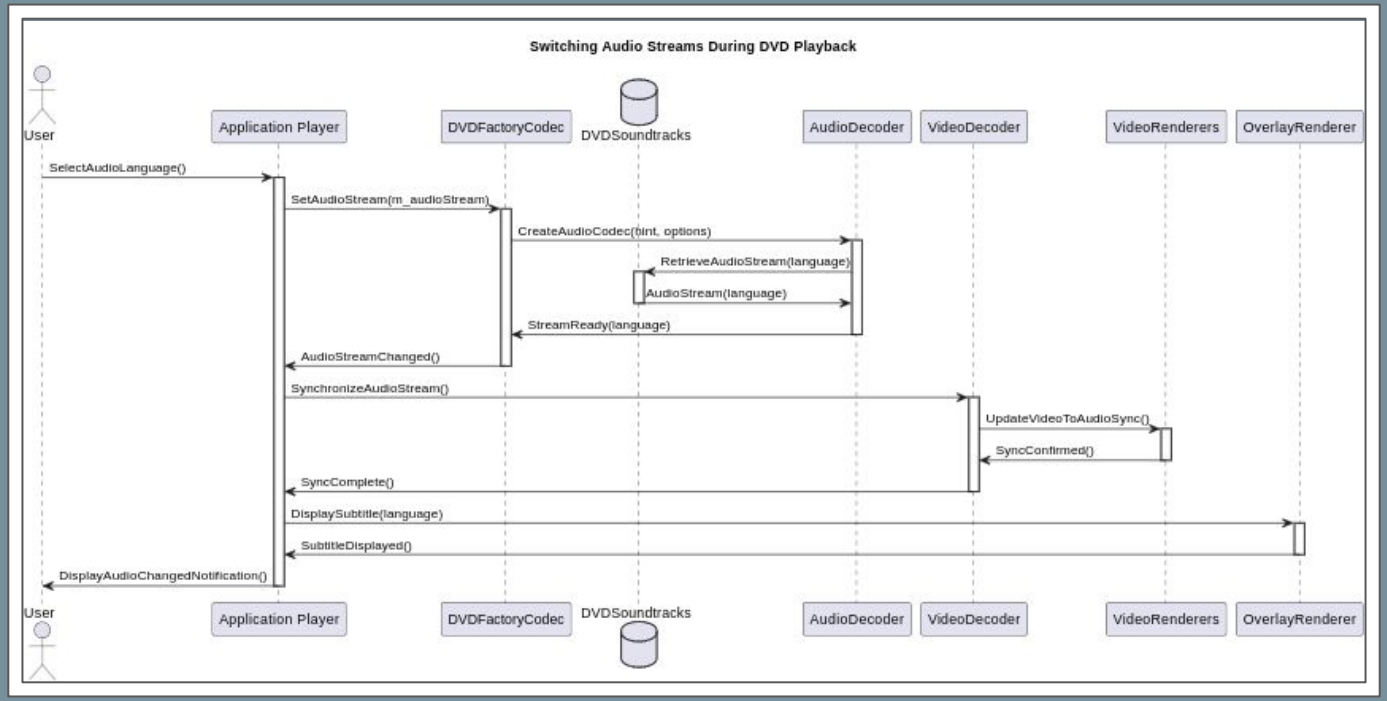
- Some components do still map to the original layers
  - However, their subcomponents interact freely with other subcomponents in other layers
- The main four components include Core Functionality, User Interface, Media Handling and System Integration

Our Derivation of Kodi's Concrete Architecture



How Did We Derive This?

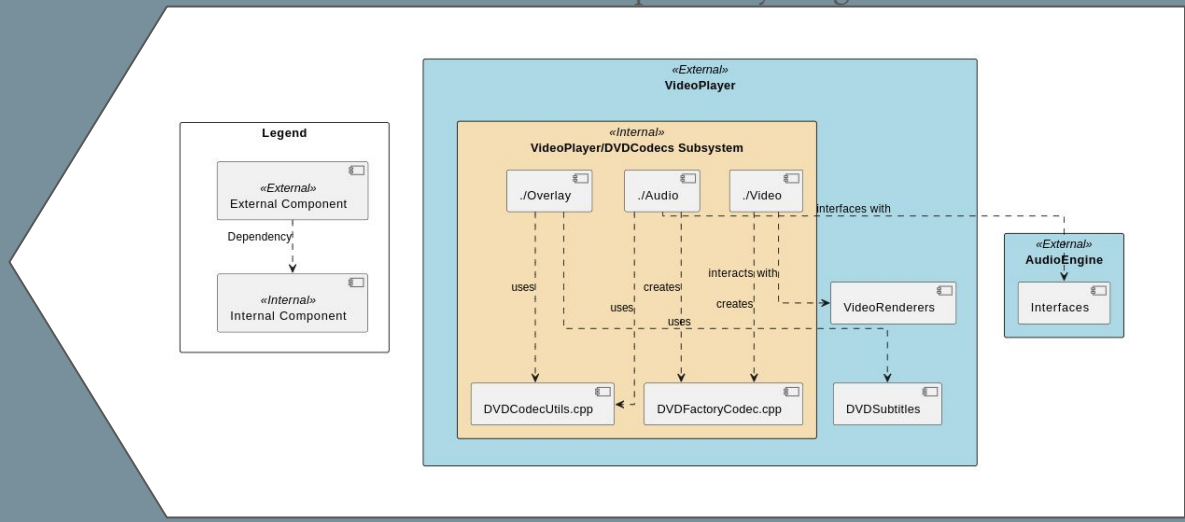
- We used Understand, a software analysis tool that turns file systems into useful visuals
- Beyond that, we also examined the file repository manually



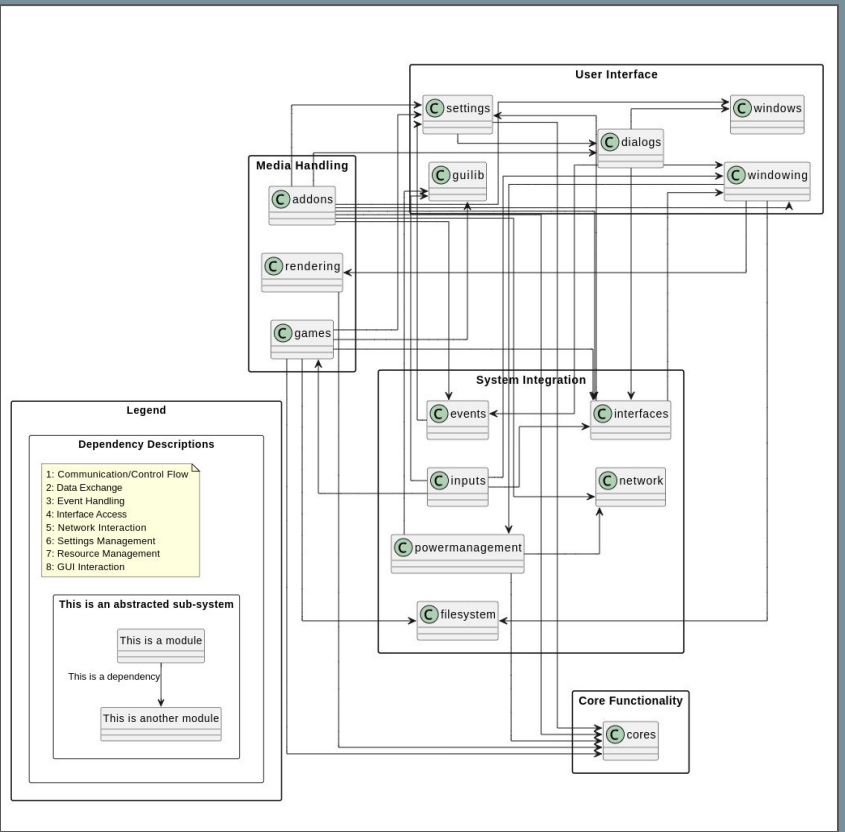
Kodi's Subsystem - DVD Player

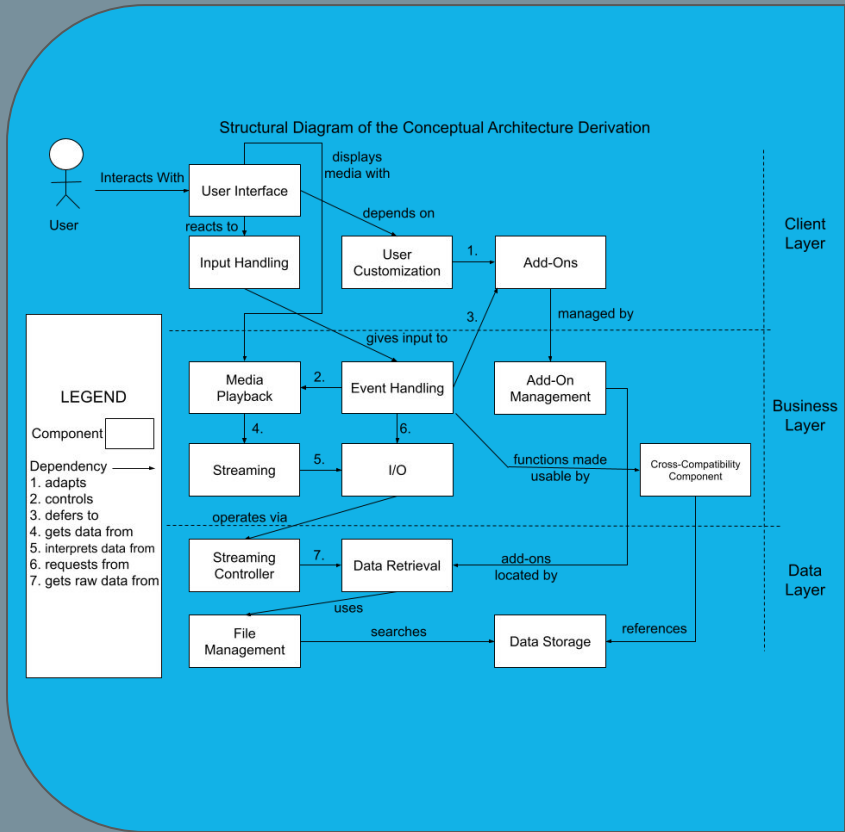
- This subsystem is responsible for handling DVD playback
- This process is split between three main subcomponents, Audio, Video and Overlay
- The many files of each subcomponent are dependant on a single file within DVD Codecs, which lead to some inefficiencies

DVD Codecs Dependency Diagram



Comparing This To Our Conceptual Derivation...





Reflecting On Kodi's High-Level Concrete Architecture

- Kodi's architecture isn't *too* far off from what it was designed to be
- However, the strict adherence to layers resulted in certain inter-layer difficulties
- The development of Kodi over time, with its many additional features, has warped the initial architectural plans into what it looks like now
    - The unexpected dependencies lead to more efficient software, and the additional features improved its utility, but that detracted from the layered architectural style the developers originally sought
    - The Client, Business and Data layers' borders have blurred

DVD Playback Subsystem Reflection

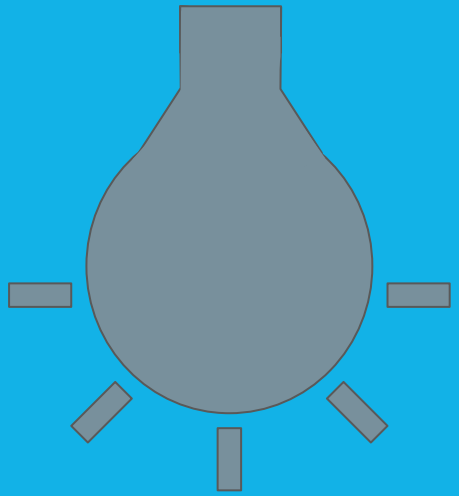
- Once more, the layered architectural design's inherent issues proved too great
  - Within the subsystem, there isn't really *any* architectural structure
- There are over 300 files dependent on one file within the main DVD Codecs section
  - This dependency structure makes the subsystem vulnerable to issues should development teams aim to edit the file being relied on
  - Alternative structures such as "pub-sub" may have fared better here

Lessons Learned

- We learned what conceptual architecture is and how it is different from conceptual architecture
- We learned how to use Understand*

Limitations

- *Or at least we started to…
- We are still not Kodi developers…



To Summarize

- Kodi's concrete architecture, the *actual* structure of the software, differs from its conceptual architecture, the *planned* structure
  - What we thought was mainly layered was more repository structured
- This was achieved through analysis of Kodi's top level architecture as well as one of its subsystems
  - That subsystem, the DVD player, suffered some inefficiencies due to how it was structured
- This showed us how important it is to implement an architecture that is more appropriate for a given software's goals