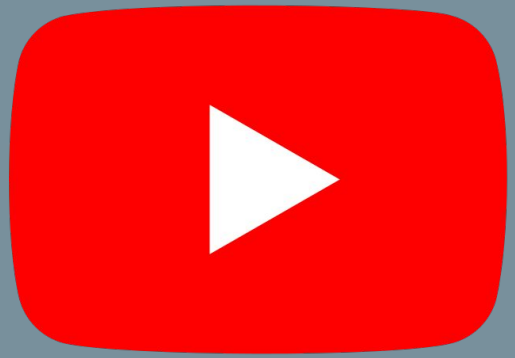


YouTube Link

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

Copy this link:

https://youtu.be/6HC44PSfWNA

or just click here!

For those who would prefer to read, the video transcript for each slide is located in the speaker notes section, found here at: https://docs.google.com/presentation/d/1SWw8AZZPTfuz_rsz3xsO4YNnoLMsBD39NSmxbaB-qCo/edit?usp=sharing

Repos and Repose: Kodi's Conceptual Architecture

Group 2 - ArchAngels

Intro, Conclusion, Abstract and Learned Lessons Writer - Bianca Bucchino: 20blb@queensu.ca
Architecture Analyst/Writer - Adam Clarke: 21asc6@queensu.ca
Presentation Designer, Narrator, Diagram Designer - Christian Higham: 20csdh@queensu.ca
Use Case Analyst/Writer - Omar Ibrahim: 20omha@queensu.ca
Architecture Analyst/Writer - Owen Rocchi: 19omr6@queensu.ca
Group Leader, Architectural Analyst/Writer - Aidan Sibley: 20ajs18@queensu.ca

Presentation Overview

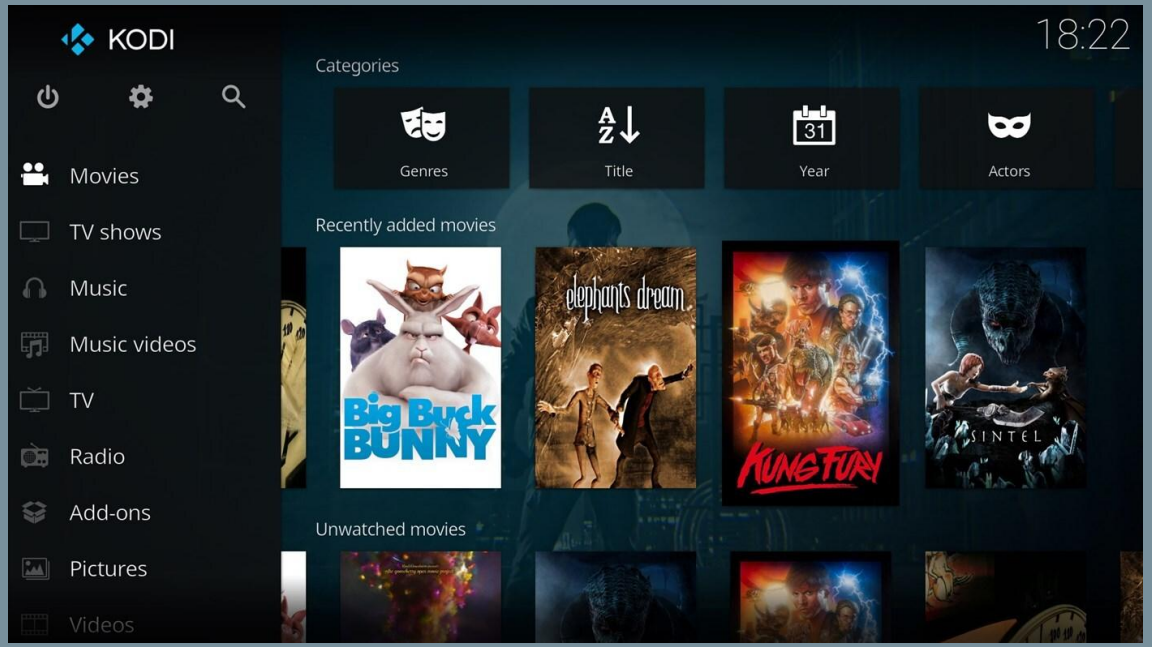
- What is Kodi?
- Derivation Process
- Conceptual Architectures
  - Models that Fit
  - How they Fit
  - Other Models Considered
- Use Cases
- Concurrency and Team Issues
- Limitations
- Lessons Learned
- Conclusion

What is Kodi?

- Kodi is an open source media player, available on many operating systems
- It plays files from a user's file system
- It has built-in support for add-ons

Kodi Logo

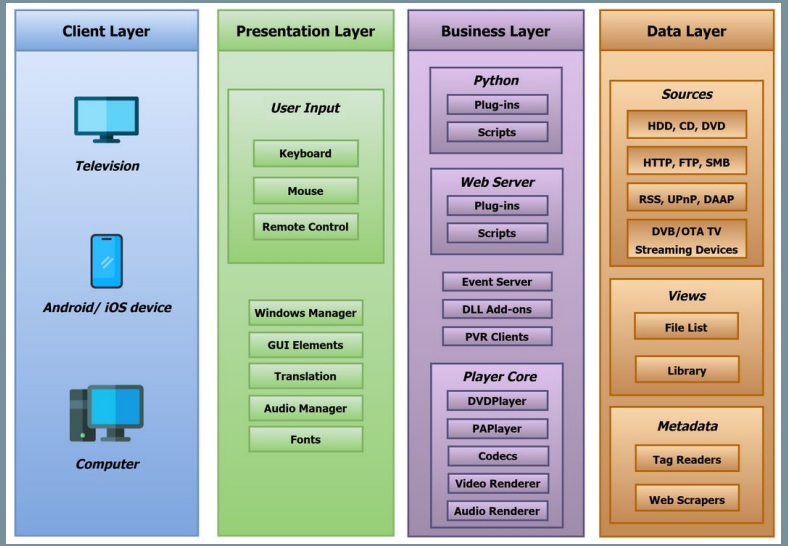
Typical Kodi Menu



https://mirror.accum.se/mirror/xbmc.org/screenshots/kodi-recently-added.jpg

Deriving Kodi's Conceptual Architecture

- The official Kodi wiki provided up-to-date architectural info
  - Layers and modules within are given simple descriptions
- Running Kodi ourselves gave us insight into how it works
- The source code of Kodi was also examined
  - Did not contribute to the analysis



https://kodi.wiki/view/Architecture

Kodi's Conceptual Architecture

Layered

- Separated into layers based on component functionality
- Limited interactions between non-adjacent tiers
- User interacts only with the top layer

Repository

- Kodi operates on the central data structure that is the user's file system
- Manipulates various files via media playback

Implicit Invocation

- Kodi itself is the central publisher
- Add-Ons are subscribers
- Kodi does not fully know how the Add-Ons will interpret its invocations

Are Other Architectures Applicable?

Maybe…

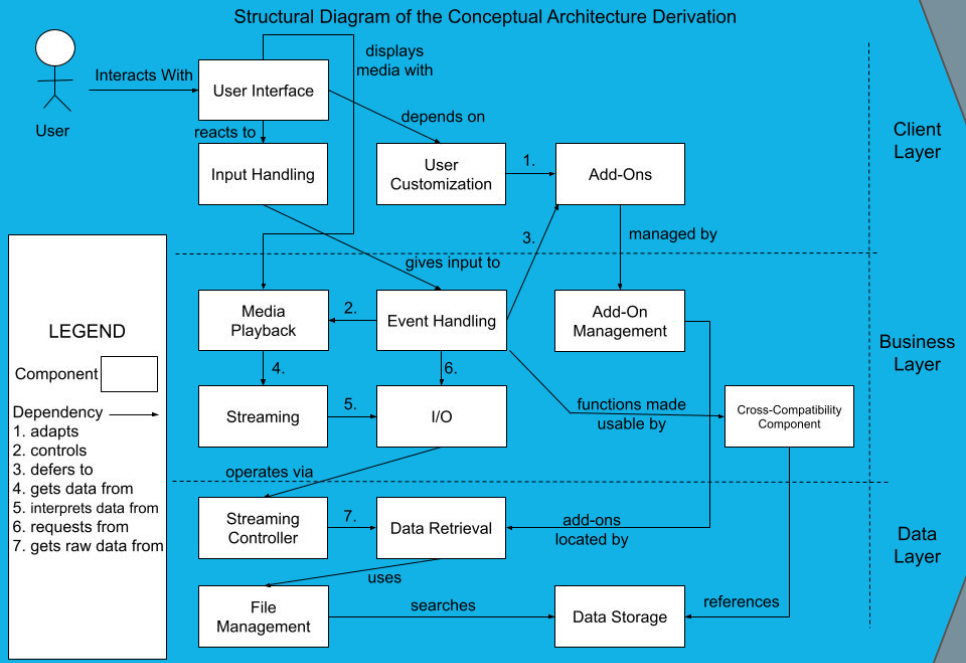
Interpreter Style?

- Interprets audio/visual files into audio/visuals via media player
- Struggles to conform to interpreter style framework
- Does not capture the scope of Kodi accurately

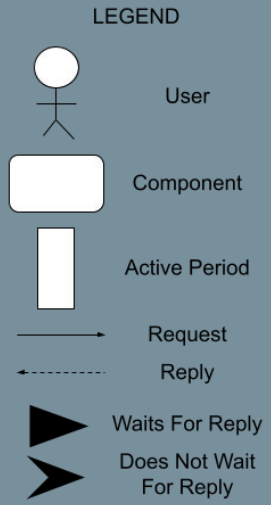
Object-Oriented Style?

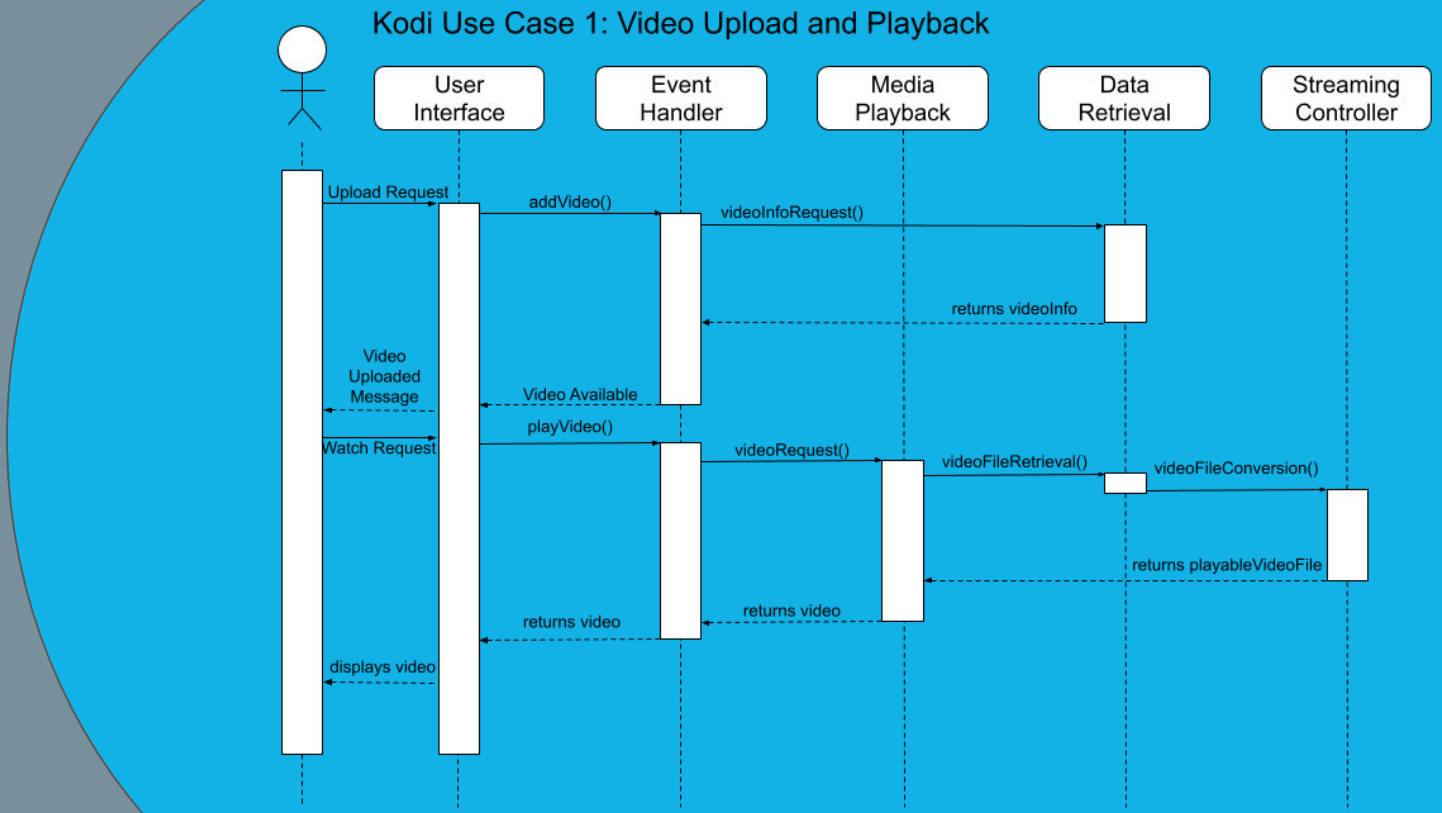
- Media files and add-ons can be considered objects
- Does not describe Kodi itself

Derivation of Kodi's Conceptual Architecture



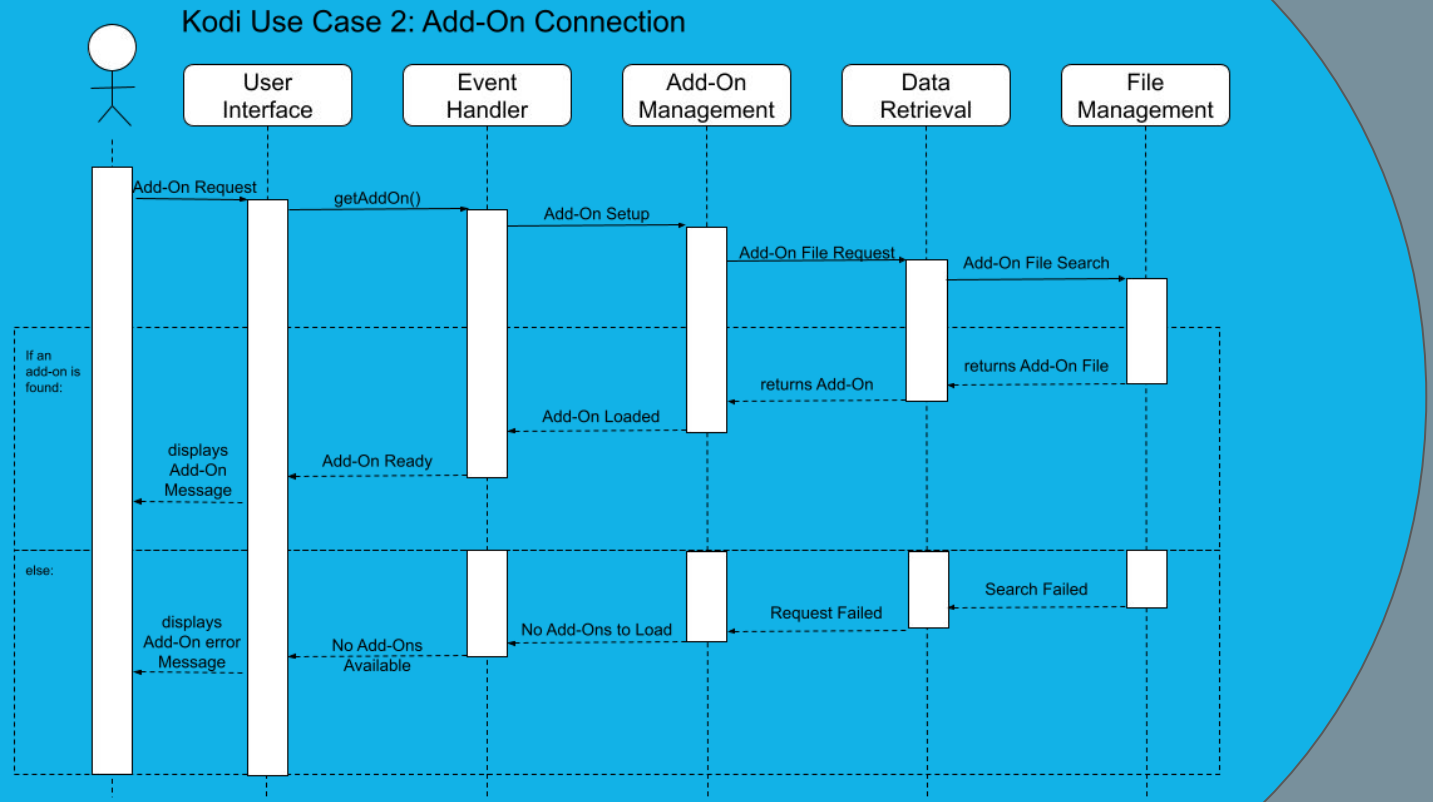
- Our derivation produced three layers, each containing many subsystems
- Their interactions are quite complex
- Lack of direct communication between subsystems on non-adjacent layers is inefficient

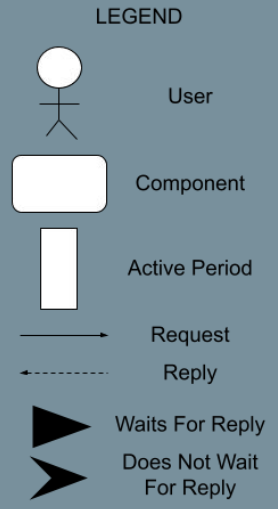




Kodi Use Case 1

Kodi Use Case 2





Concurrency

- User Interface/Input Handler always run, concurrently
  - These in turn may trigger other modules, such as the Event Handler. The Event Handler also runs concurrently with its called processes

Team Issues

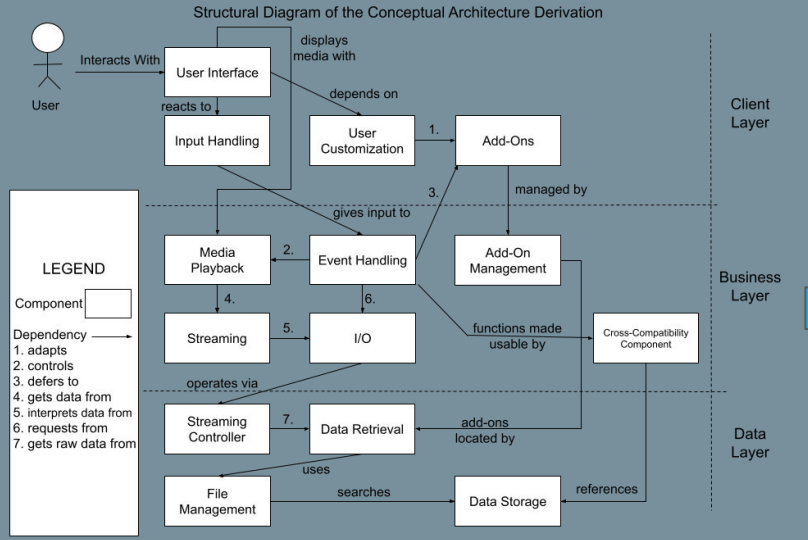
- The rules of layer interactions may lead to inefficiencies during runtime
- Add-Ons may conflict, producing unexpected results; pub-sub makes this hard to prevent or counteract

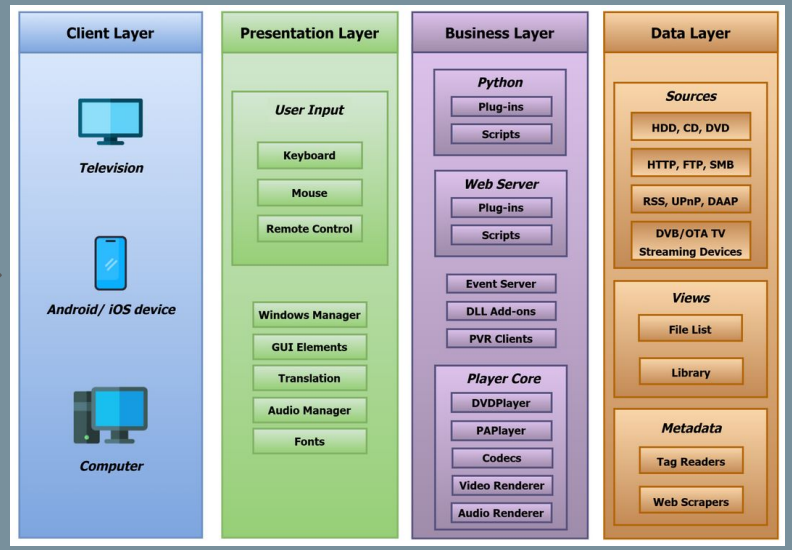
Limitations

- We are not Kodi's developers
- Derivation process is based on current architecture
    - We are extrapolating the conceptual from the concrete
    - This may result in mistakes

Lessons Learned

- How conceptual architecture applies to a finished product
- Matching architectural styles to end goals
- General importance of organizing software





https://kodi.wiki/view/Architecture

In Summary

- Kodi, the multi-platform media player software, has a layered architecture with supporting repository and pub-sub style systems
  - Kodi interacts and manipulates data from a device's file storage repository
  - Kodi's add-ons are implicitly invoked, for better or worse
- Through the segregation of modules via layers, a great amount of concurrency occurs, while also resulting in some inter-layer inefficiencies
  - Best exemplified through use cases
- Reflecting on Kodi's architecture revealed the inherent flaws and advantages architectures have for given software objectives