CISC 322
A1
Kodi - Conceptual Architecture
Date: October 22nd, 2023

Authors:
Ricardo Chen (Team Leader) – 21rc20@queensu.ca
ZhouJun Pan (Presenter) – 19zp9@queensu.ca
Jiahao Ni(Presenter) – 19jn35@queensu.ca
Lucy Zhang – 19ytz@queensu.ca
Shuaiyu Chen - 21sc33@queensu.ca

Table of Contents:

Abstract

Kodi is a prominent open-source media player that serves as a foundation in the field of multimedia consumption. This report delves deep into the structure and design of Kodi, highlighting its intricate built-in C++ source code of a vast number of lines and thousands of files. Our goal is to examine the significance of Kodi as a robust platform that goes beyond the limits of conventional multimedia players. In the rapidly evolving era of multimedia players, Kodi flourishes in its media environment, where the requirement for versatile, compliant media solutions is ever-present. Kodi's user-friendly interface, which enables users to easily access and handle a diverse range of media formats on a wide range of systems and devices, highlights the architectural complexity of the software. The goal of the report is to give a comparative architectural assessment of Kodi that will help programmers, researchers, and media player fans learn a lot about how this great player works. Kodi represents the ideals of open-source software, exhibits the capability of collective cooperation among developers, and most importantly, illuminates the tremendous possibilities and opportunities of distributed media consumption in present-day society. For all the passionate Kodi users and those with growing enthusiasm for open-source technology, this study paper presents the internal mechanisms and architecture that highlight its value.

Introduction

Kodi is a highly versatile and powerful cross-platform media player with access to a wide variety of multimedia activities for users worldwide. The XBMC Foundation, a non-profit technology company, created this free and open-source software. Today, it has received contributions from hundreds of enthusiasts and developers. It is a testament to open-source development and an example for other applications as well. The active community involved in enhancing its features has made it possible for Kodi to provide the seamless experience that it provides to its users.

The development process of Kodi has an interesting history. Originally known as the Xbox Media Player (XBMP), it was developed for the Xbox game system to transform gaming platforms into entertainment hubs. After its success, it went through a significant rebranding and was ultimately known through its own entity called Kodi (Dreef, et al., 2015). We live in

a time of unrestricted multimedia consumption, and Kodi offers a distinctive platform for controlling various media types, simplifying this experience. Kodi has a user-friendly interface that puts personalization right at your fingertips with a variety of media, including movies, music, television episodes, and images. Both casual users and multimedia enthusiasts can easily use this platform. This wide range of functionality and customization, as well as support for major media formats like MP4, MKV, AVI, MOV, and FLAC, is what makes Kodi popular (Rev, 2022). It also has streaming capability from websites like YouTube, Netflix, and Hulu (Kamen, 2017). This signifies that the company has been inclined towards progressive developments.

Kodi has many other features, like library management, where media files from local storage can be added to the user's library for categorization and easy future use. It also has another useful feature: metadata scraping. It allows users to scrape metadata, like movie posters and record covers, to enhance the aesthetic attractiveness of the user's media library (Paul, 2009). Through its third-party extension, users can access a wide range of other features, like custom themes and visuals, play games, access weather, and many more (Fitzpatrick, 2009). Kodi is available on all of the major platforms and operating systems. This makes it even more versatile, contributing to its popularity. Users can customize the program to suit their preferences and add functionalities.

Apart from its user interface and popularity, let's talk about its internal mechanism and how it was made. Kodi was made using a popular programming language, C++. It's a low-level language, highly versatile, and has low-level access to system resources, allowing for direct connection to hardware components and making it fast and reliable. The source code of C++ is in millions of lines, which indicates the level of sophistication. Thanks to its super-technical capabilities, users can browse a variety of media platforms. With C++, Kodi is designed with a modular architecture. This supports the creation of reusable and efficient code and adds functionalities and add-ons as required (What is kodi…, 2022). While C++ is platform-dependent, its machine-independent feature allows it to work on different computer systems and is incredibly fast, allowing Kodi software to deliver high-performance media playback.

Another major strength of Kodi is its open-source nature. This makes Kodi unique among traditional media players. The open-source nature of Kodi brings several advantages for both programmers and users. The majority of development in Kodi is done not with the intention of profit but with user accessibility. The volunteers have continually contributed to enhance and keep it up to date with cutting-edge technology and user needs. Open source, as the name suggests, means that the source code of Kodi is publicly accessible. So, any developer can edit, modify, find issues and bugs, and contribute to the software. This gives transparency and security to users, as they can inspect code freely. It also promotes innovative thinking among several developers and users. This way, anyone who thinks of adding features or removing issues in software can act as they wish. Overall, users receive dependable and current software, and Kodi is certain to benefit from a wide range of developers.

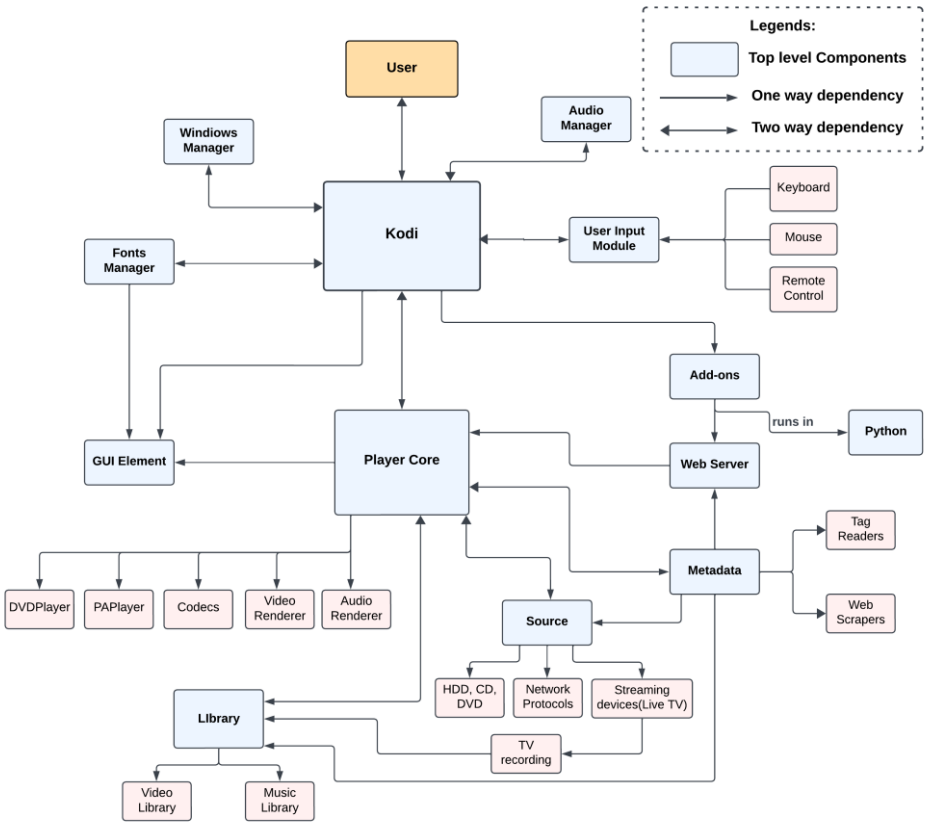
Kodi is completely free for all users around the world. It doesn't have any hidden subscriptions or licenses. As it is not motivated by money, the application is not made to be addictive in order to earn more revenue as well (Kodi Wiki, 2023). Rather, the software is only focused on usability, scalability, and security.

For this report, we downloaded Kodi to mark our journey. We watched a movie to

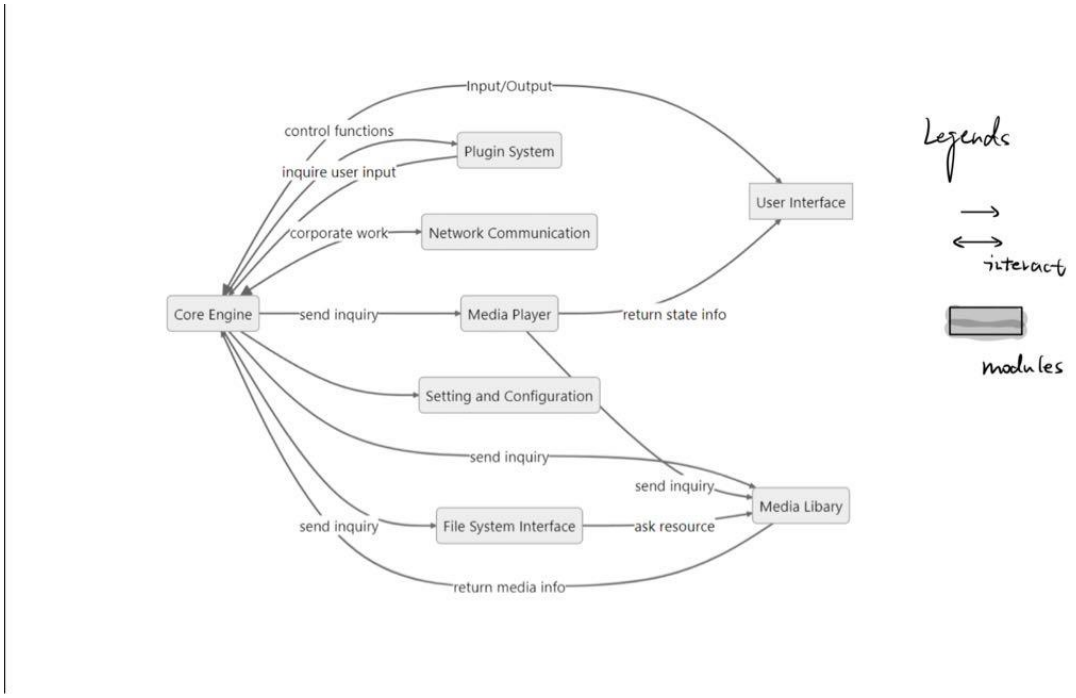
experience the real-life interface of Kodi and were delighted with its user-friendly features. Afterward, we explored the complex architecture of Kodi. We researched the complexity of its components, design factors and layers. Kodi architecture is in 4 layers, just like most of the popular applications: client, presentation, business and data. The client layer is for handling user input and presenting the user interface. The presentation layer is for managing multimedia playback and showcasing the user interface. The data layer is for storing and retrieving data and library files. Finally, there is the business layer, as the backend layer is for core functionality management and add-on support.

The report acts as a compendium of Kodi, providing insights into its architecture, design, and relevance. It is an invaluable source for software developers, academic researchers, and media enthusiasts with an interest in multimedia working mechanisms. Kodi, as an open-source media player, has completely changed the way in which we browse multimedia and interact with it. It is one of the best examples of the power of community-driven creation and a future example of open-source development for other applications. The paper investigates the inner working mechanism of Kodi, its architecture, user interface, and effect on the world of multimedia consumption. Kodi has changed the way multimedia players run and will continue to advance with the help of contributors all around the world.

Architecture



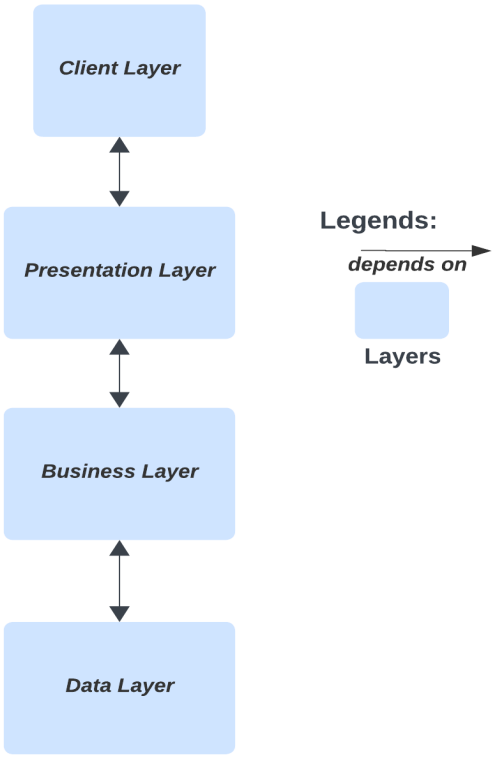
Kodi's architectural style is a modular and layered architecture style. Based on modules, different functions are divided into independent modules or components. Each module is then responsible for a specific task. Secondly, Kodi adopts a layered structure and divides different functions according to levels. There are several layers such as user interface layer, core engine layer, service layer, media library layer and media playback layer. In Kodi, the core engine is the top priority of Kodi, managing media playback and user interaction, and is responsible for coordinating all components. And when processing user input, it provides the main control loop, which facilitates routing it to the corresponding subsystems and modules. Second is the user interface. The user interface is the user's main point of interaction. In Kodi, the user interface interacts with the core engine, responds to user input, and performs actions based on user needs. Then Kodi's user interface is modular, and users can customize different looks and layouts according to their preferences. Then there is the media library, which is used to handle large amounts of media files and metadata and interacts with the file system interface to scan and index media files, extract metadata, and organize them into a structured database. At the same time, the media library subsystem also interacts with the core engine to meet users' media browsing and search needs. The media player is responsible for playing media content. It interacts with the core engine and helps users control playback. The plug-in system also interacts with the core engine for plug-in management, and with network communication components to interact with external servers. Network communications interact with the core engine to integrate external services. The file system interface interacts with the media library to scan and index files. And interact with the media player to play files. Settings interact with subsystems, allowing users to customize the behavior of Kodi to suit their needs. The key parts to Kodi's performance are the media player and media library. Among them, the media player needs to ensure real-time playback, and the media library needs to optimize the index in a large media library to meet performance requirements. For future changes, Kodi's addon system is a great functional component that allows functionality to be added by interacting with external servers. And Kodi's modular architectural style can help Kodi adapt to changing user needs. In Kodi, the core engine is used as the central control point, managing the flow of control based on user needs.



Kodi's architecture primarily follows a modular architecture style. It divides the software into distinct and independent modules or components. This modularity makes it easier to maintain and extend the system. The architecture emphasizes cross-platform compatibility, allowing Kodi to run on various operating systems and hardware configurations. This flexibility supports the system's goals and requirements of providing a rich multimedia experience to a diverse user base.

Kodi's modular architecture and support for add-ons (Python Modules) enable future changes and enhancements. Third-party developers can create add-ons to extend Kodi's functionality, ensuring that the system can evolve and adapt to changing user needs and preferences. Additionally, the modular design allows for easy updates and maintenance without affecting the entire system, enhancing its evolvability and testability.

Components Breakdown and Interaction



Kodi's presentation layer is dedicated to user interface and interaction components. Its main function is to manage the graphical user interface, user input, and visual aspects of the application. This layer contains key components such as user input modules, Windows manager modules, and GUI elements. The user input module captures user input from various devices and sends it to the graphical user interface to enable user interaction. Kodi relies on the underlying operating system's window system for window management, and the Windows Manager module ensures that this interaction is seamlessly integrated. GUI elements include various visual components such as windows, buttons, dialog boxes, and lists that together form a user interface for navigation and interaction. These components within the presentation layer work together to provide Kodi users with a user-friendly and visually appealing experience, allowing them to easily interact with the application and its content.

The business layer is a critical component of the Kodi architecture responsible for managing server access, handling external libraries, enabling custom add-ons and extensions, and ensuring smooth playback of multimedia content. It acts as an interface between users and multimedia content. The main purpose of this layer is to read, display, and play multimedia content in high quality while facilitating interaction with external services and add-ons.

The business layer encompasses key components such as the Python Module, which empowers users to enhance their Kodi experience by creating custom add-ons and extensions.

Additionally, the Web Server Module plays a crucial role in allowing remote content management via web browsers and other applications, ensuring that multiple users can access multimedia content over a common network. The Player Core Module, a critical component within the Business Layer, manages multimedia content playback, handling various multimedia formats, synchronizing audio and video, and controlling playback flow to provide a seamless and high-quality viewing experience. Together, these components in the Business Layer enable Kodi to interact with external services, play multimedia content, and extend its functionality through add-ons.

The data layer in Kodi is dedicated to content management and storage, ensuring efficient access and organization of multimedia content from various sources. This layer encompasses key components, including Sources Elements, Views Elements, and Metadata Elements. Sources Elements manage and organize diverse media sources, allowing users to access content from local storage drives, network protocols like HTTP and FTP, and even streaming devices such as DVB. Views Elements focus on how multimedia content is presented and organized, offering various views such as file lists and libraries to enhance the user's browsing experience.

Metadata elements play a crucial role in enriching the user experience by obtaining additional information about multimedia content. This includes the use of Tag Readers to retrieve metadata from local files in the library and Web Scrapers to retrieve metadata from online sources. The data layer works together to ensure that multimedia content is efficiently managed, organized, and presented to users in a user-friendly and informative manner.

In Kodi, various interactions work together to provide a user-friendly experience. The user input module captures user actions and passes them to the user interface, allowing the user to interact with the media player. GUI elements create a visual interface and display content options. User input is sent to business layer components such as the Python module for add-ons, the web server module for remote access, and the Player Core module for content playback. The Player Core module connects to the web server module to retrieve and stream content from various sources. Metadata elements, such as web scrapers, add additional information to improve the user experience. These interactions make Kodi a user-centric multimedia system.

Evolution

The evolution of Kodi reflects the continuous development and open nature of a multimedia center system, striving to provide users with an enhanced media experience. The project's inception can be traced back to the early days of Xbox Media Player, which took root between 2002 and 2003 when it was initiated by two software developers, d7o3g4q and RUNTiME. Initially, these developers embarked on separate projects named XboxMediaPlayer and XBPLAYER, which later merged into one project.

As time passed, another software developer, Frodo, joined the Xbox Media Player team, facilitating the amalgamation of XboxMediaPlayer with the YAMP project, giving birth to the Xbox Media Player 2.0 project. This integration marked the confluence of multiple independent efforts and projects into a more extensive framework designed to meet user needs comprehensively.

Nonetheless, the development of XboxMediaPlayer was short-lived, yet it served as a pivotal steppingstone for the progression of XBMC. This phase signified the birth and growth

of the XBMC project, laying a robust foundation for future multimedia centers.

On December 13, 2003, the development of the Xbox Media Player was halted to prepare for the initial launch of the Xbox Media Center. On June 29, 2004, XboxMediaCenter 1.0.0 was officially released, signifying XBMC's transition from a mere media player to a versatile multimedia center. This transformation marked the point at which XBMC no longer confined itself to media playback but expanded its focus to encompass the diverse multimedia needs of users.

In the subsequent phases of XboxMediaCenter's development, the project gradually introduced an array of features. In October 2004, version 1.1.0 was released, adding support for more media, files, and container formats and video playback, karaoke support, and other functions. Version 2.0.0 was released in September 2006, introducing RAR and zip archive support, a new audio/music player (PAPlayer), DVDPlayer upgrades, iTunes 6.x DAAP, and UPnP client support. With the continuous release of versions, XBMC continues to improve and add new features. In 2007, the XBMC Media Center software was first ported to Linux, and in 2008, the XBMC Media Center software was ported to Mac OS X and Windows.

2010 marks XBMC entering a new open era. The release of XBMC 10.0 Dharma introduces an add-on system that enables users to easily customize the XBMC experience and converts core content into plug-ins, enabling automatic updates. This version also includes improvements to other important features such as hardware acceleration. The XBMC team first ported the software to iOS in 2011, followed by the first port of the XBMC Media Center software to Android in 2012. The open-source nature of XBMC allows it to be continuously expanded and improved to provide users with a more powerful media center experience. The release of XBMC 13.0 Gotham in 2014 brought numerous improvements, including hardware decoding on Android, speed improvements on Raspberry Pi and Android, stereoscopic 3D rendering support, improved touch screens, and subtitle search improvements. At the same time, extended Python and JSON-RPC APIs for developers provide greater potential for future development.

On July 31, 2014, the XBMC announced an important decision: the system was renamed Kodi. This rebranding aimed to eliminate confusion with the Xbox and emphasize Kodi's multifaceted capabilities. The subsequent release of Kodi 14.0 Helix in 2014 further enhanced the software by upgrading FFmpeg to support efficient video codecs, empowering users to play high-efficiency video formats. This version also introduced user control over add-on updates and customizable keyboard layouts, particularly benefiting tablet and remote-control users. In 2015, Kodi 15.0 Isengard brought a remarkable development as it was officially launched on the Google Play Store, making it more accessible to Android users.

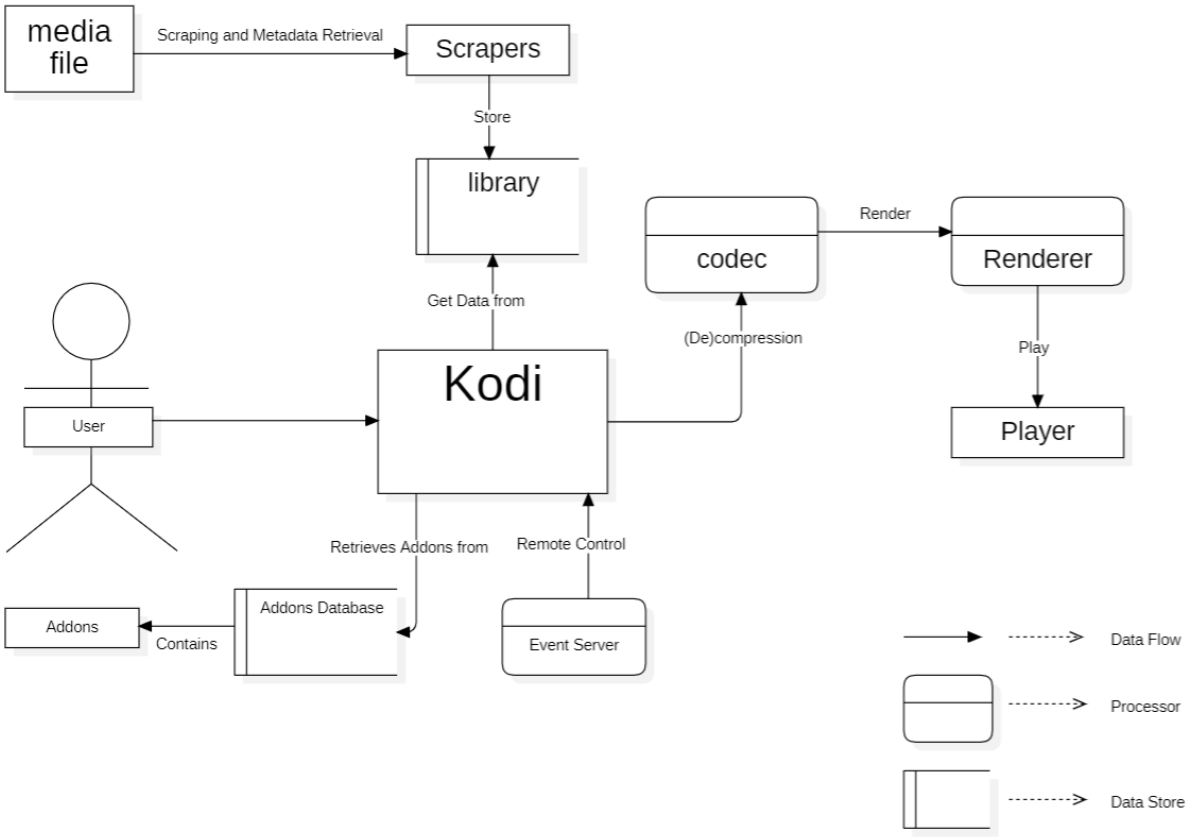
Entering 2016, Kodi 16.0 Jarvis - Mark XVI was released, which introduced the event recording function, allowing users to track Kodi activities and add-on updates. In 2017, Kodi 17.0 Krypton was released, bringing users an updated user interface, improved audio and video engines, and an upgraded audio API for the Android platform, adding support for more advanced audio formats. In 2019, Kodi 18.0 Leia was released, which enhanced support for game emulators, ROMs, and controllers. In 2021, the Kodi 19.0 Matrix was released, which improved the audio and video playback experience, improved the user interface and skin, added subtitle options and migrated Python plug-ins.

By 2023, Kodi 20.0 Nexus was released, introducing binary add-on support for multiple

instances, significantly improving the subtitle system, implementing game Savestate support, adding Windows HDR support, adding NFSv4 support, and improving context menu consistency, to improve system stability, performance, and security.

Currently, Kodi 21.0 "Omega" is in the pre-release phase, paving the way for new possibilities in Kodi's future development. This continually evolving media center system consistently strives to provide users with an exceptional multimedia experience, featuring a broad range of features and continuously improving performance.

Control and Data Flow



As it shown in the figure. The central component of Kodi is responsible for user interaction. Users communicate to Kodi with the operation of UI. To extend the functionality of Kodi during run-time; Kodi makes use of Add-ons which are created by the third-party developers. It retrieves addons from Addon Database and does so from a remote control. The Kodi is also connected to an event server that handles events from the users such as playback start/stop, pause/resume, etc. The event Server listens for commands from event clients. Anything that can communicate using UDP can be an event client. Many event client software packages are currently available for PCs, Macs, smartphones, PDAs, and more. Some event client software accepts commands from infra-red remotes, gamepad controllers, and others to translate and send to the EventServer

Kodi gets some other resources from the library, which provides access to various types of media content such as movies, TV shows, music, etc. Where do these files come from. It relies

on a functional component called scrapers, which are responsible for scraping metadata from various sources such as IMDb, TMDb, etc. For example, when users measure to add resources via URL, the scraper will retrieve information about the media files from URL and add them to the library

One of the main data flows is how Kodi play media filesWhen you want to play or edit these files, the codec will decompress them to make them available for playback or editing. The renderer is responsible for decoding and converting compressed video data into visual images. Finally, the video will be played on the player.

Concurrency

As we know, concurrency in software architecture refers to the ability of a system to perform multiple tasks at the same time without having to wait for one task to end before proceeding to the next. Simply put, it refers to the ability to handle multiple tasks within the same time period. For concurrent implementation, multi-threading, parallel computing and other methods are usually required. And issues such as concurrency control and data sharing need to be considered. This has a lot to do with the operating system. We will conduct a concurrency analysis of Kodi mainly in terms of multitasking.

As far as we know, Kodi is a media player for single users. It does not support simultaneous use by multiple users. Because the existence or implementation of concurrency needs to be considered, we focus on the main core functions of multi-user access, multi-user remote collaboration or multi-user data sharing. But for the media player Kodi, personal use on local devices is the main business function. Therefore, we conclude that there is no concurrent operation capability for Kodi.

But as far as we know Kodi is an extensible software. Users can add plug-ins to achieve functions that Kodi itself cannot achieve. Therefore, users can achieve the function of concurrent use by multiple users by installing plug-ins. For example, MySQL is a database sharing plug-in that can help multiple users access the same media library. The web interface plug-in allows users to access Kodi's interface through a browser, which means multiple users can collaborate remotely and concurrently.

In general, concurrency is beneficial to improved system performance, more efficient resource utilization, and faster response times. However, whether a system requires concurrency depends on the nature of the system, user needs, performance requirements, and hardware and software environments. But in terms of current computer usage, concurrency has gradually become an essential design functional requirement.

External Interfaces

Kodi relies on a network of components and external interfaces to provide its users with a seamless multimedia experience. While users may not need to be concerned about the precise form of data storage, understanding the conceptual architecture of Kodi's external interfaces is necessary to appreciate its functionality and versatility.

Graphical User Interface (GUI): The Graphical User Interface (GUI) is at the heart of the

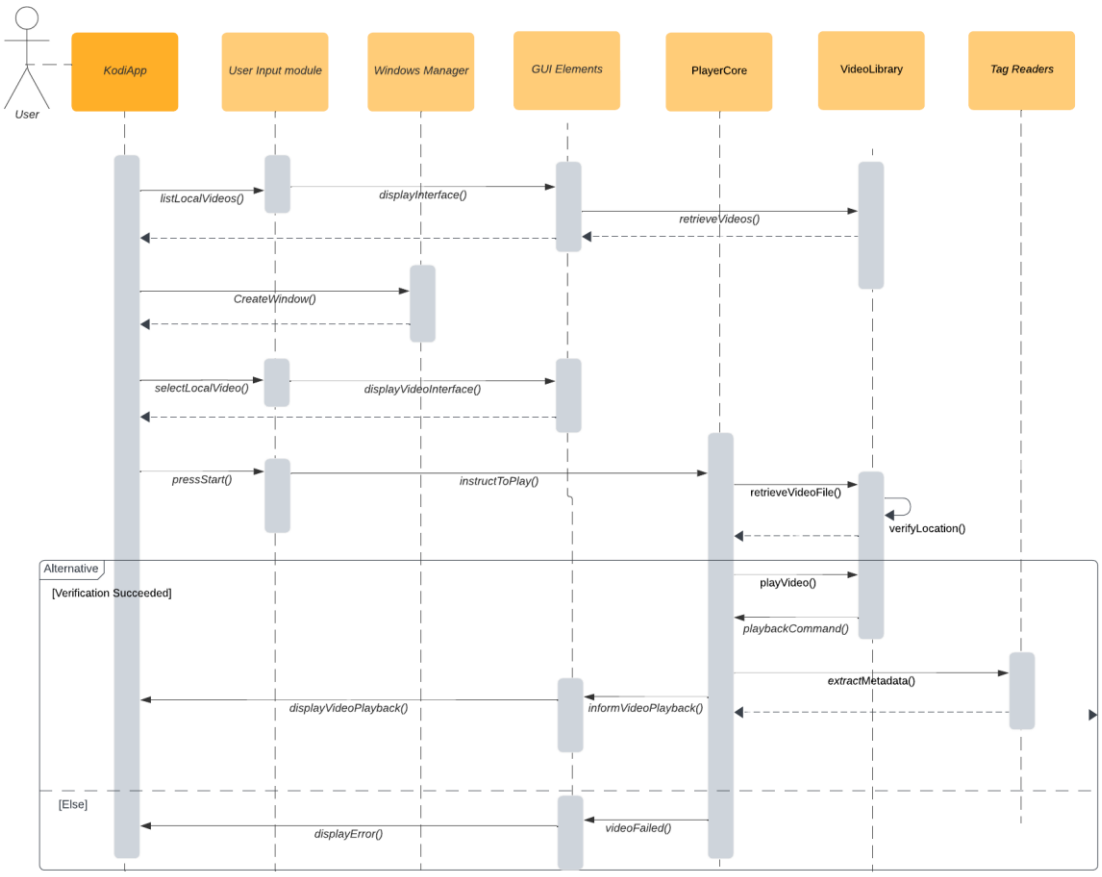
Kodi experience. The graphical user interface (GUI) provides users with an easy-to-use platform for sending and receiving multimedia content. It provides information about the user's media library, playback status, and personal preferences, making it easier for users to navigate the Kodi ecosystem.

Application Programming Interface (API): Kodi includes an API that enables third-party applications and services to interact with the platform. Third-party developers can now create plugins, extensions, and integrations as a result of this.

Database: In Kodi's architecture, the database primarily stores critical metadata, such as information about media files, such as titles, genres, and cover art. It ensures a well-organized and user-friendly media library.

Use Cases

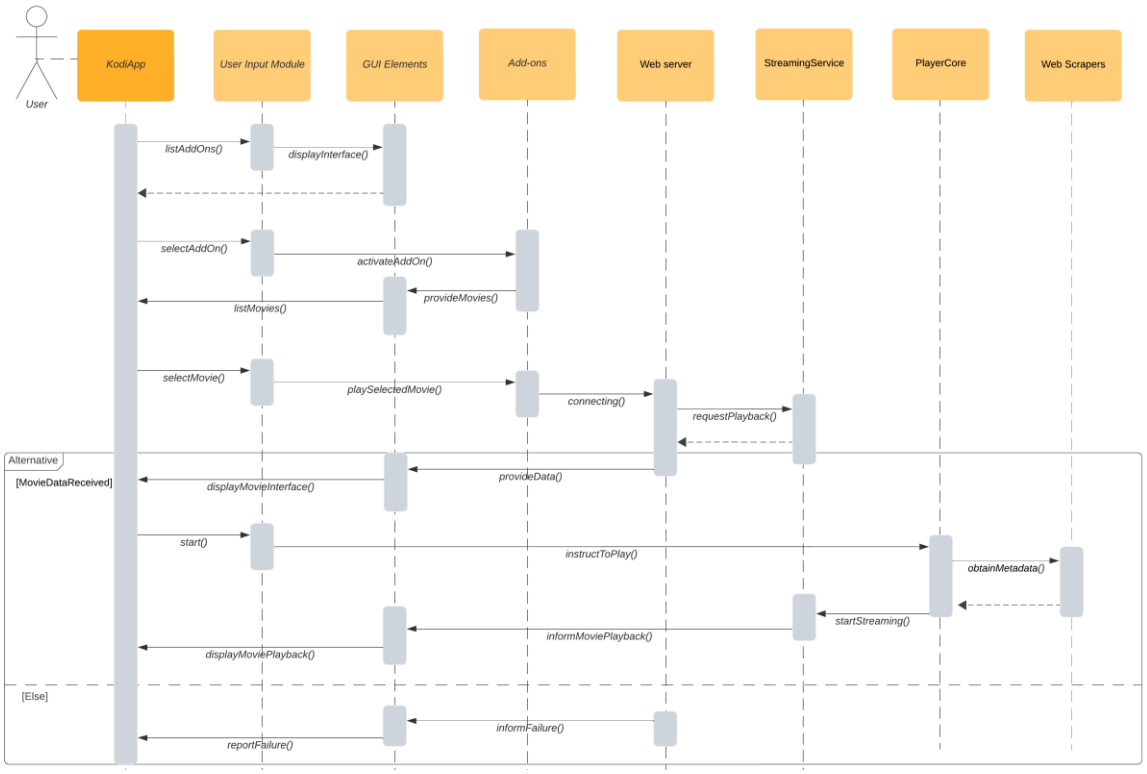
Use Case 1: Playing a local video



When the user chooses to play a local video, the Kodi application is first launched, and the user browses and selects the video file through GUI elements, then retrieves the video from the video library. This triggers the core media player, which decodes and plays the video. Users can use playerCore to control the playback behavior of the video according to their preferences

and needs.

Use Case 2: Watching Online Movies



When a user watches an online movie, start the Kodi application first and use the GUI to select a movie add-on, such as Netflix, or YouTube. The Kodi application connects to the online streaming service through Add-ons and Web Server modules. After the user selects and plays a movie, the Streaming Service module obtains and transmits the movie streaming data, which is decoded and played by the PlayerCore module, while the Web Scrapers module provides movie data. These ensure that users can watch online movies smoothly.

Data Dictionary

The presentation layer is the system's front end, and it includes the user interface and user interaction components. It is in charge of rendering the graphical user interface, handling user input, and managing the application's visual aspects. This layer's components may include GUI Elements, User Input Modules, and Windows Manager Modules, among others.

Naming Conventions

Modular Architecture: An architectural approach that divides a software system into distinct, self-contained modules or components.

Cross-Platform Compatibility: The capability of software to run effectively on various operating systems and hardware configurations.

Add-Ons (Python Modules): Extensions or plugins developed by third-party developers to enhance Kodi's functionality.

Front End: The user-facing part of the software that manages user interactions and displays information.

Conclusions

In conclusion, Kodi's architectural design demonstrates a well-thought-out and well-structured foundation for its multifaceted multimedia capabilities. Through a well-defined separation of concerns among different layers, the architecture of Kodi promotes modularity, scalability, and reusability. This not only improves security but also allows for the independent development and maintenance of individual components, making it easier to add new features and resolve issues.

Lessons learned

The lessons learned from Kodi's open-source journey highlight the remarkable power of community-driven development and the significant benefits of transparency in the realm of software development.

Despite the potential benefits of Kodi, there are limitations to its use that need to be addressed:

1) Legal issues: The legality of Kodi remains largely ambiguous as no clear laws have yet been made against the streaming of copyrighted content, almost nowhere in the world. When users try to use scraper to scrape metadata from the internet, some resources are forbidden in their countries

2) Add-on risks: Kodi’s add-ons are not always safe and can pose security risks to users. Some add-ons can contain malware or spyware that can harm your device or steal your personal information

Upon examining the limitations of Kodi, it is vital to explore the critical lessons that its architecture and implementation provide

1) Application core: Since Kodi follows the layer style, the core is the key of this system. This component provides the basic functionality of Kodi, such as media playback, user interface, and add-on management.

2) evolvability: Kodi’s architecture is designed to be flexible and modular, making it easy for developers to add new features and functionality. For example, to be able to extend the functionality of the application core, Kodi allows uers to manage add-ons which are developed by third-party developers.

3) Documentation: Due to the open-source feature, Kodi is well-documented on

some website, such as Kodi offical website, Kodi wiki, and Github. It literally helps users to master the basic use of Kodi and understand the development process and architecture.

References

Ars Technica. https://arstechnica.com/information-technology/2009/12/xbmc-911-makes-your-open-source-home-theater-look-shinier/

Development. Official Kodi Wiki. (n.d.) https://kodi.wiki/view/Development

Dreef, K., Reek, M. van der, Schaper, K., & Steinfort, M. (2015, April 23). *Architecting software*

to keep the lazy ones on the couch. Kodi. https://delftswa.github.io/chapters/kodi/

Fitzpatrick, J. (2009, April 5). *Customize XBMC with these five awesome skins*. Lifehacker. https://lifehacker.com/customize-xbmc-with-these-five-awesome-skins-5198009

Kamen, M. (2017, May 3). *What is Kodi and is it legal? A beginner's guide to the home media server*. WIRED UK. https://www.wired.co.uk/article/kodi-how-to-beginners-guide

Kodi Wiki. Official Kodi Wiki. (2023). https://kodi.wiki/view/Main_Page

News. Kodi. (n.d.). https://kodi.tv/blog/

Paul, R. (2009, December 29). *XBMC 9.11 makes your open source home theater look shinier*.

Way Back Machine. (n.d.) History. Way Back Machine https://web.archive.org/web/20080217040527/http://xbmc.org/about/history/

What is kodi media player? all you need to know. TechOwns. (2022, December 12). https://www.techowns.com/what-is-kodi/