Master's thesis 2021

Development of a Reference Architecture for streaming of Cloud infotainment system to In-Car Thin clients

Arjun Krishna Murthy
Ramkumar Venkatesh

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Development of a Reference Architecture for streaming of Cloud Infotainment system to In-Car Thin clients

Arjun Krishna Murthy Ramkumar Venkatesh

Supervisor: Christian Berger and Michel Chaudron, Department of Computer Science and Engineering
Advisor: Patrik Andersson, Aptiv Contract Services AB
Examiner: Gregory Gay, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Abstract

The gaming industry is rolling out new gaming solutions, where games are hosted and rendered in the cloud and streamed to thin clients in 60 fps with 4K resolution. This technology enables new possibilities in extending the lifetime of existing hardware, simplifying software deployment to multiple clients, and enabling data analytics. These are topics that are of great interest to the automotive industry, where vehicles are typically in use for ten years or more. By providing solutions similar to the gaming industry, the automotive sector could address these issues. The expectation is that car users should be able to get the same gaming experience in an infotainment system as with playing it on the phone. But, it is not possible currently due to certain limitations. The video streaming from the phone to the car head unit requires devices in the head unit. Even having such a device in head-unit will not guarantee a high-quality, low latency video gaming experience because of lag issues, as when you stream your phone screen, a lot of operations are involved. Using the design science research methodology, we elicited the functional and non-functional requirements required to develop and design the architecture through semi-structured interviews. Based on the data collected from the interview and the literature study, AGL's technical feasibility for real-time low latency content streaming was reviewed. Architectural patterns and tactics were used to design the AGL architecture, which supports streaming in both online and offline mode. Architecture evaluation using a semi-structured interview was adopted to evaluate the functional and quality attributes identified for the architecture developed.

Keywords: Computer, science, computer science, software engineering, AGL, In-Vehicle infotainment system, project, thesis.

Acknowledgements

The authors of this thesis would like to thank Aptiv AB, specifically Mr. Patrik Andersson, for being incredibly supportive and motivating in every aspect of this thesis. We would like to sincerely thank the pillar of support to our thesis, our college supervisors, Mr. Christian Berger and Mr. Michel Chaudron, for guiding us throughout the project. Finally, we would like to express our gratitude to all the respondents for the support and willingness to spend some time with us to participate in the interviews.

Contents

List of Figures

1 Introduction

The automotive industry faces competition as there is a rapid increase in the development of electric and connected vehicles leading the companies to deliver flexible products and be used in the long run. Recently there are various innovations that have started to evolve, leading to the advancement of technology. These innovations are meant for enhancing the safety of the vehicle driving on the road, optimizing the performance of the engine, managing different forms of in-built automated systems, and offering infotainment.

In-Vehicle Infotainment (IVI) system is a feature that combines both software and hardware by providing vehicle information and entertainment to drivers and passengers. By 2022, the infotainment system will see a tremendous increase in its usage [11]. The smart vehicles currently enhance in-car user experience through multimedia-enabled infotainment systems [67]. The intelligent vehicles are expected to carry communication systems, computing facilities and require high storage and sensing power. A number of solutions were proposed to address these challenges, and Vehicle cloud technology is one of the solutions. Recently, there are advances in cloud technology, where the focus in not on computation alone. Cloud technology offers high definition rendering, which turns the idea of cloud gaming into a reality. Cloud technology is used in the gaming industry where cloud gaming in its simplest forms, renders an interactive gaming application remotely in the cloud and streams the scenes as a video sequence back to the player over the internet [3].

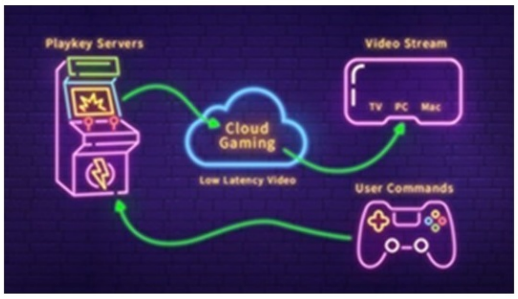


Figure 1.1: Cloud Gaming [79]

The software in the automotive industry is populated with commercial tools and applications due to safety and security concerns. In recent years, there are efforts to make the software open and customizable through the open-source community.

With the emergence of vehicle-to-vehicle communication and vehicle-to-cloud technologies, the developers are switching to open source and portable standards to achieve interoperability between different vehicles. Some of the infotainment platforms used by automakers are QNX, GENIVI, Windows Embedded Automotive and Android Automotive. According to Höttger [70], AGL (Automotive Grade Linux) was identified as the most appropriate car platform over Apertis, Ubuntu Core, QNX, and Android Automotive. Automotive targeted open-source operating systems such as Automotive Grade Linux (AGL) [68] is a shared platform mainly used for In-Vehicle infotainment system used mostly by the automakers and suppliers to reduce fragmentation and to reuse the same code base leading to rapid innovation, and faster time-to-market [69].

The automakers' major issue for implementing connected cars is that each manufacturer uses custom operating systems to develop In-Vehicle Infotainment systems. Thus, there is no standard platform being followed for the development of IVI systems [69].

Currently, in the automotive industry, as seen previously, there are limitations in streaming high content, low latency video from phone to infotainment. Additionally, AGL does not provide a clear roadmap of technical requirements about accomplishing near real-time, low-latency, high data volume content streaming in their future releases [44]. These limitations lead us to ponder over a possible solution which can be inbuilt into the car infotainment system, where the near real-time, high-content low-latency video is streamed, rather than relying on external devices for streaming.

Another exciting technology that is in the development stage is Holoride [49], which gives hyper-immersive experiences by combining navigational and car data with X Reality(XR). X-Reality is a form of mixed reality environment that is a union of wide range of sensor/actuator networks and shared online virtual worlds. It includes wide variety of hardware and software, including sensory interfaces, applications, and infrastructures that helps in enabling content creation [50]. It connects standalone XR devices to the vehicle and is pre-installed. This technology is in the alpha stage, but we can see that it comes pre-installed. But if it can be streamed from cloud to the car, then the XR experience given can be updated regularly and provide a much better experience to the changing user needs. To overcome these problems, we see a need for research using cloud-based technology for the In-Vehicle infotainment systems.

There are several advantages when using cloud technology for an In-Vehicle infotainment system. Cloud technology for infotainment systems helps in extending the hardware life for the infotainment systems. It also simplifies the software development for the In-Vehicle Infotainment systems as most of the applications are housed in the cloud. Cloud technology also helps to improve the data analytics where most of the data is stored in the cloud, and it is easier for the automakers or suppliers to understand the behavior of the system and also helps to run the IVI system efficiently. From a business perspective, the cloud solutions for infotainment systems provide a high-end user experience with low-cost hardware and also helps in managing infotainment systems in a cost-effective way.

1.1 Background

The Automotive Grade Linux (AGL) is a shared open-source platform used mainly for In-Vehicle infotainment systems by the automakers and suppliers to reduce fragmentation and reuse the codebase, which will lead to rapid innovation and faster time to market [69]. Automotive Grade Linux (AGL) provides APIs at all levels from CAN (Controlled Area Network) to Infotainment by improving entertainment and safety on board [71]. Automotive Grade Linux (AGL) acts as a reference framework based on which the OEMs and suppliers manufacture their own products by integrating their own innovations. One of the important goals of Automotive Grade Linux (AGL) is to provide a full user interface access to OEMs where automakers and suppliers rely on a single open source In-Vehicle infotainment software platform [69].

Reference architecture captures the essence of the architecture of a collection of systems [72]. The purpose of the reference architecture is to act as a guide that can be used for future developments [15]. It can be considered as an architectural pattern for developing an architecture for software systems of a particular domain. reference architecture helps to systematize and standardize the development of software systems and helps to reuse the design [16]. Establishing reference architectures is an important issue, as they can describe both essential building blocks as well as design choices for dealing with functional and non-functional requirements in cloud environments [5]. According to Angelov et al. [6], how reference architecture can be evaluated by applying existing evaluation methods meant for concrete architecture is assessed. Challenges in applying the existing evaluation method to reference architecture are identified by them, and they recommend how the evaluation methods can be applied to reference architecture with certain changes.

1.2 Problem Domain and Motivation

In-Vehicle infotainment (IVI) requirements are more complex than any other consumer electronic device [13]. When implementing features similar to Personal Digital Assistant (PDA), smartphones, etc., for the IVI units, they need to provide a seamless user experience without interfering with the actual driving, which is the main and most important task when using a vehicle. According to a study, removing eyes from the road for just two seconds or longer doubles the risk of a crash [46]. The IVI units must be working across the lifetime of a vehicle, which is 5 to 10 times longer than any smartphone [13] [14]. Some of the factors which contribute to the reason that most car head units are not up to the mark with the latest smartphones, are as follows. Firstly, cars usually have a long lead time. The time between the initial car design phase and final delivery of the car from the manufacturer with various features or particularities is technically 1-2 years. By the time the car reaches the customer, it will include an infotainment system, which may be outdated. To get the latest infotainment systems or to update the units, the customers need to approach the car manufacturers. Secondly, the components which are included in an IVI system need to withstand vibration, dust, extreme temperatures, and moisture, which the smartphones are not subjected to(many be not on a similar level of IVI system). The innovation cycle between car makers and vendors will be slowed down when the car makers use off-the-shelf consumer products as the products will not be of automotive-grade. Lastly, car makers are teaming with tech companies (Apple and Google) as they realize that it is difficult to develop the feature in the car to improve user experience.

At present, the car manufacturers allow certain applications that are present on smartphones to be accessed on car infotainment by connecting the phone to a car infotainment system, by providing support for Android Auto [31] and Apple CarPlay[32]. The number of applications that we can use from a phone in the car head-unit is limited by platform owners like Google and Apple. If the restriction to limit the application by platform owners is removed, we should be able to access applications from the phone in the car IVI. If we connect the phone via USB to IVI, we should be able to stream the content directly or be able to play Google Stadia games on the infotainment system. But, it is not possible currently due to certain limitations. The video streaming from the phone to the car IVI requires the Chromecast device[43] (incase of Google Stadia) in the head-unit. Even having a Chromecast device in head-unit will not guarantee a high-quality, low latency video gaming experience because of lag issues, as when you stream your phone screen, a lot of operations are involved. An alternative solution to reduce lag is to use an HDMI cable to connect to Infotainment system [42]. HDMI provides a bandwidth capacity of up to 10.2 Gbps, more than twice the bandwidth needed to transmit an uncompressed 1080p signal, resulting in better-looking movies, faster gaming, and a richer audio/video for consumers. In order to use the advantages of HDMI in infotainment, support to HDMI port becomes an additional hardware requirement. This will require changes to be made to both phone and car infotainment hardware, which incurs overhead costs.

In the current roadmap of AGL [44], they have mentioned certain Vehicle-to-Cloud (V2C) scenarios, which include data logging, GPS tracking, remote control, and diagnostics, but do not provide details of technical requirements about accomplishing near real-time, low-latency, high data volume content streaming.

It can also be noted from the literature that cloud technology focuses on seamless streaming of applications and also high-level models for developing the applications. In fact, there is a lack of consensus on which functional elements and non-functional properties must be addressed by platforms targeting cloud-based systems[5]. Also, to the best of the authors' knowledge, there has not been much research conducted in developing an architecture for the automotive infotainment system which uses cloud technology.

1.3 Research Goal and Research Questions

In this thesis, we investigate an open-source platform, which is Automotive Grade Linux (AGL), and explore how cloud applications like Spotify, Google Stadia stream high content low latency data and the technical limitations associated with the application when used on the car platform. We also look into the software platforms Apple CarPlay and Android Auto, which duplicates the user smartphone, and also the technical specification required by Spotify when it is used in CarPlay and Android Auto. We also investigate the technical specification of audio and video components currently supported by the AGL to stream cloud-related high content data. With the analysis performed, we create a base for the architecture, and based on the base created, we design and develop the reference architecture which supports a high-quality, low latency real-time content streaming of infotainment from the cloud to the car during online and offline mode [1]. The reference architecture is also assessed on robustness and generality to ensure architecture fitness to purpose [2].

The purpose of this thesis is to address the above issues mentioned in the problem domain section, leading us to define the following research goal:

"Understanding the issues involved in enabling cloud-based near real-time, low-latency, high data content streaming to car platforms which provide high end-user experience."

Based on the above research goal, the research questions are as follows:

RQ1: What are functional and non-functional attributes that are required for the reference architecture to support near real-time, low-latency, high data volume content streaming?

RQ2: How can we evaluate that the attributes identified in RQ1 have been satisfied in the reference architecture developed?

RQ3: How can the AGL reference architecture be extended to support near real-time, low-latency, high data volume content streaming?

1.4 Contributions

This thesis will explore the current features offered by AGL and the features proposed in its roadmap. This could potentially result in possible feature additions that can be made to AGL.

The study will investigate the working of software applications like Android Auto, Apple CarPlay, Google Stadia, and Spotify through a literature survey. It will identify the current requirements and limitations that exist in these applications for streaming data from Cloud to vehicle. The researchers will carry out a design science research methodology in order to understand AGL reference architecture and provide suggestions to expand the reference architecture to support near real-time, high volume, low latency data content streaming.

This thesis would add to the existing body of academic knowledge within software engineering when it comes to designing reference software architecture and tactics used in the context of streaming near real-time, high volume, low latency data from cloud to vehicle. Furthermore, the study contributes with the knowledge of identifying the functional and non-functional attributes that are required when designing the architecture for cloud content streaming to vehicles.

The thesis will also serve as a good pre-study resource for researchers interested in designing reference architecture for streaming near real-time data from cloud to vehicle, as they can find some insights from the thesis.

1.5 Scoping

To focus on the investigation of the characteristics mentioned in this thesis, we decide to explicitly exclude legal aspects involved in content streaming in-car platforms and focus only on the technical aspects of content streaming. Additionally, we reduce the scope to focus on the infotainment side (client side) rather than the cloud side (server side).

1.6 Structure of thesis/paper

This thesis is structured as follows: in section 2, we present the background. In section 3, we discuss the related work. In section 4, we describe research methodologies, followed by section 5 that explains the results of our research, and in section 6, the main analysis and discussions of results are presented. In section 7, the thesis ends with a conclusion and future work.

2
Background

This section will give a brief about Automotive Grade Linux(AGL), Android Auto and Apple CarPlay, and Spotify.

2.1 Automotive Grade Linux (AGL)

Automotive Grade Linux(AGL) was launched by the Linux Foundation in 2012, which was developed as an open-source project to help the automakers to rely on a Linux-based platform. AGL provides a basic software kernel based on Linux OS and also includes several software modules like communication, web services, graphics, navigation, and automotive services. The AGL provides Unified Code Base (UCB), which includes an operating system, middleware components, and application framework. This will provide 70-80% of the starting point required for the development of the IVI system. The remaining 20-30% is for the automakers and suppliers where they can customize it according to the customer requirements [69]. The latest AGL Unified Code Base (UCB) release 9.0.3, named Itchy Icefish, was released in August 2020 [73].

The main concept associated with AGL is that it will act as a reference framework which can be used and supported by the OEMs and suppliers to deliver their own customized product by integrating their own ideas and innovations. To do this, AGL aims to provide automakers and suppliers a common platform that provides full user interface access. In this way, AGL can achieve its goal of developing and implementing a fully open-source IVI for vehicles at a rapid pace. The In-Vehicle Infotainment system is the communication point of information in a connected car. The services and functions of the car are controlled by the vehicle's driver, which utilizes the IVI. The IVI system also supports route planning and also provides communication of roadside services. The Automotive Grade Linux (AGL) focuses on standardizing such open-source IVI [68].

2.1.1 AGL Software Architecture

The AGL software architecture is represented by four layers, application/HMI layer, followed by application framework layer, followed by service layer, and lastly, operating system layer. The application/HMI layer supports different features that are displayed to the users. The application framework layer, also called the system layer, provides the APIs for application development and communication. The ser-

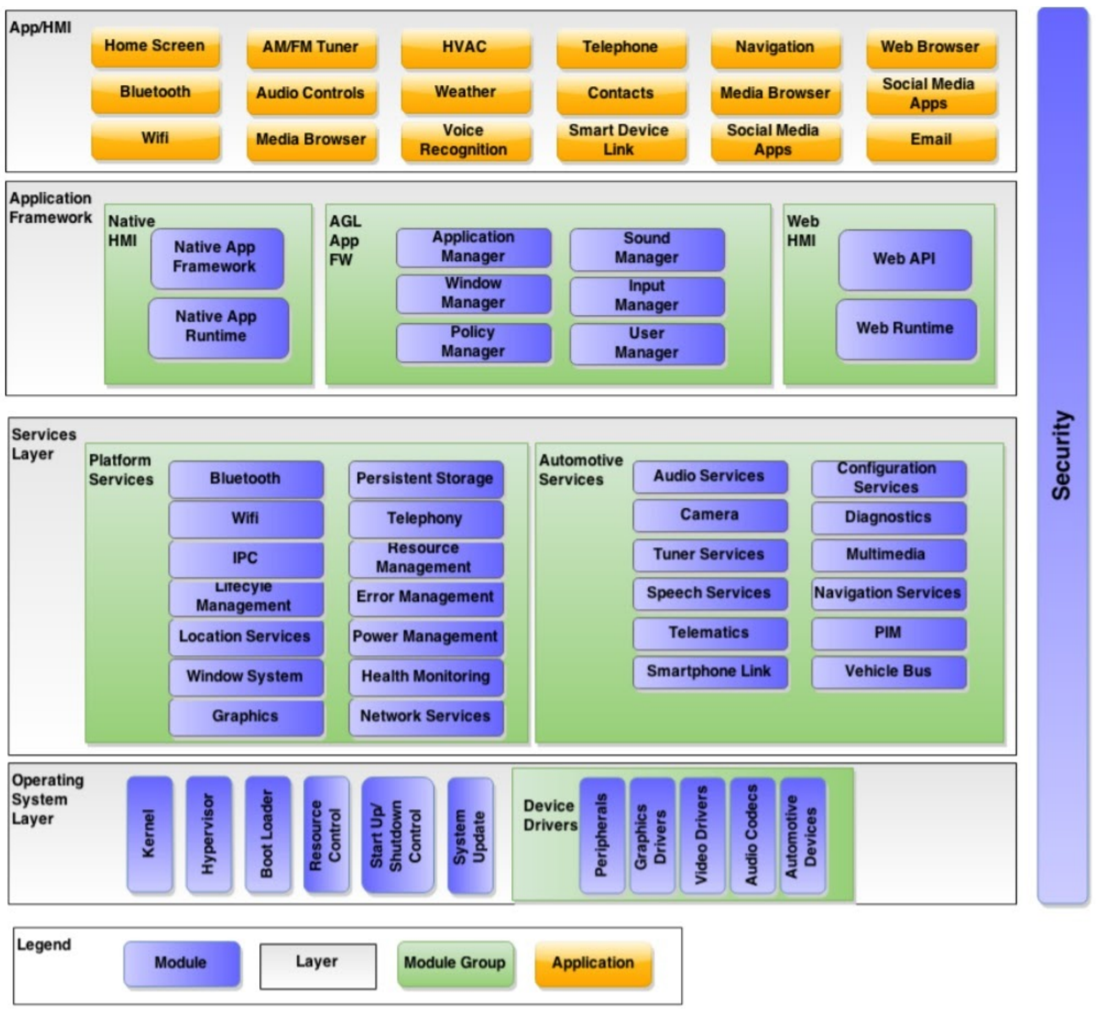


Figure 2.1: AGL Software Architecture [80]

vice layer contains all software available user-space resources. Lastly, the operating system layer includes the kernel, different device drivers, and common operating system utilities. The user interface and functionality of the AGL framework are made entirely in JavaScript and HTML5, and the platform communicates with the car via an Automotive Message Broker (AMB) using Tizen IVI web runtime to allow applications to transmit data to and from the vehicle. The features which are currently included in AGL's user experience are Home Screen, Dashboard, Heating Ventilation, and Air Conditioning (HVAC) system, Google Maps, Internet and News Service, etc. [69]

2.2 Android Auto and Apple CarPlay

Apple-designed CarPlay as a multimedia technology for passenger cars, and it serves as co-pilot[33]. The users can perform many operations like navigation feature, listen to songs, read incoming messages and reply to them, answer incoming calls using the Siri feature without driver's attention-getting distracted. The voice control button on the steering wheel can be pressed and held by the user to enable synchronization between CarPlay compatible multimedia devices and iPhone to activate the voice control feature. It can be used by touch sense in the touchscreen of vehicles.

Google developed Android auto as a smartphone projection standard to control the smartphones from vehicles. The USB interface is used by Android auto to display Google Now interface on the car's info screen. Users can use the car's touch screen, steering controls, and other control arms while using Android Auto. The Android Auto interface can only support the approved applications that meet Google's security requirements.

But, Google and Apple, in the role of the platform owner, determine which smartphone applications are unlocked for use in cars[34]. The developers get access to necessary APIs to adapt the functionalities for the application in cars. In order to minimize the distraction to the car driver, certain adjustments are necessary, as restrictions are imposed on applications that can be used in cars; for example, display applications should be in reduced form on the infotainment screen. Additionally, animations are prohibited for applications in cars, which could cause a distraction to the driver. The use of CarPlay or Android Auto in cars always requires the agreement of the OEM, which decides about the integration of the solutions provided by Google or Apple. The applications (on the smartphone), which are certified by the platform owner (Apple or Google) can be used through the head-unit. The selection of applications and its design of graphical user interface (GUI) is the responsibility of the platform owners, as mentioned previously. The GUI of the platform of the respective OEM is deactivated as long as the smartphone integration is active. Hence, the number of applications that be used on car head-unit is currently limited by the platform owners and OEM agreements. Additionally, there is a minimum hardware requirement of head-unit for Android Auto, and CarPlay [35] [36]. For instance, CarPlay to work with car head-unit, requires the head-unit to have a minimum physical display of 6 inches, 24-bit color, 30 Hz refresh rate, support for H.264 video decoding, high-resolution display is required - 16:9 aspect ratio (800*480, 1280*720, 960*540, 1920*720 pixels), USB infrastructure. As a result of the above requirements, a list of car manufacturers that support Android Auto/Apple CarPlay does not include all the cars in the market [40] but is a growing list.

2.3 Spotify

Spotify is one of the popular digital music services from which you can listen to millions of songs across many genres. Spotify is an application where users can browse songs from various playlists and listen to them. Spotify uses client-server access and peer-to-peer protocol for streaming songs from the cloud. The Spotify protocol is a proprietary network protocol designed for streaming music [17]. For streaming, clients avoid downloading data from servers until there is a need to maintain the song quality. The client, at most times, tries to stream from a peer-to-peer network.

The system requirements for using Spotify and accessing Spotify content through the Spotify app are as follows. For operating systems like iOS, it requires iOS 12 or above, Android OS 4.1 or above, Mac OS X 10.10 or above, Windows Desktop and laptops running Windows 7 or above [47]. For optimum performance, Spotify recommends having at least 1GB of free memory on your device [48]. Spotify supports various audio quality(bitrate options) to suit various devices, and plans [46]. For Spotify to work on the phone, the options are as follows. For Spotify free users, low-quality audio is equivalent to approximately 24kbit/s, normal quality is approximately 96kbit/s, and high quality is 160kbit/s. For Spotify Premium users, in addition to the free user option, we get an additional very high-quality option, which is equivalent to approximately 320kbit/s.

3 Related Work

This chapter aims to present previous work that relates to the research goal of this master thesis. Research papers, articles, and blogs regarding the automotive infotainment platforms, and specifically Automotive Grade Linux (AGL), were searched for in academic databases using keywords such as "Automotive Grade Linux" and "AGL". The reference articles were searched by using the snowball sampling technique. A systematic mapping study was conducted, outlined in the following sections, along with the summary of the AGL articles. Section 3.1 describes the systematic mapping study process, and section 3.2 provides the summary of the relevant papers.

3.1 Systematic mapping study

A systematic mapping study [74] was conducted according to Petersen et al. [74] and is described in this section.

3.1.1 Define research questions

The scope of the study is defined, which is the first step of a systematic mapping study. In this thesis, the scope was defined using the research goal and research questions in Section 1.3.

3.1.2 Search

The second step of the mapping study was to search for related articles. In order to avoid narrow search, four different online libraries were used to search for related work: ACM Digital Library [75], Springer Link [76], arXiv [77], IEEE Xplore [78]. By using Google Scholar, results from other libraries like Research Gate and Google patents were also found.

Automakers' major issue in implementing the connected car concept is that they do not follow a standard platform for the development of Infotainment systems because every vehicle line within each automaker uses customized operating systems from different suppliers. Due to this, time to market and manufacturing costs are high. By switching to an open-source solution, people can try out the software without any effort, and developers can learn various technologies without the required training. The open-source helps to customize the technologies and reduces the time-to-market. The automakers can reduce the maintenance cost by teaming up with the community behind the open-source platform. Source codes can be effectively aligned according to user needs and improved code quality and stability. Hence in this thesis, we chose AGL as an appropriate platform as it was the only open-source platform to develop IVI systems.

The initial search string used was "AGL" and "Automotive Grade Linux" and then the results were separated by setting different inclusion and exclusion criteria defined in section 3.1.3. As this search engine did not yield much of related articles, more strings were used to fetch. The final keywords used for the search were Linux foundation, connected car, open-source platforms, AGL Connectivity, Infotainment connectivity, connected vehicles, wireless connectivity, vehicle communication, Internet of Vehicles, Automotive Grade Linux AND OTA update, AGL AND OTA, AGL AND Over the Air update, AGL AND infotainment updates, AGL AND Software update, Automotive Grade Linux AND streaming, AGL AND Streaming apps, Infotainment streaming, automotive-grade Linux AND stream, automotive-grade Linux AND network AND stream, AGL streaming application.

Initially, the terms carplay, Apple CarPlay, Android Auto, Connected cars, Mobile platforms, Mirrorlink were not included. Although as infotainment platforms were studied further, it was found that those terms were used to denote platforms used by Apple and Android, which replicates the smartphone on the car infotainment system, and they were thus added to the search string.

This step resulted in 120 articles across all databases at the time of this thesis.

Search Terms	Papers found
AGL+Cloud+connectivity: Keywords used: AGL, automotive grade linux, Linux foundation, connected car, open source platforms, AGL Connectivity, Infotainment connectivity, connected vehicles, wireless connectivity, vehicle communication, Internet of Vehicles.	32
OTA Keyword used: Automotive Grade Linux + OTA update, AGL + OTA, AGL + Over the Air update, AGL + infotainment updates, AGL + Software updates	23
Streaming Keyword used : Automotive Grade Linux + streaming, AGL + Streaming apps, Infotainment streaming,Automotive Grade Linux + streaming, AGL + Streaming apps, Infotainment streaming,automotive grade linux + stream,automotive grade linux + network + stream,automotive grade linux + data + stream,automotive + infotainment + stream, automotive + infotainment + data, AGL + Stream, AGL+Infotainment, AGL streaming application	33
Apple Carplay & Android auto Keywords used: carplay, Apple CarPlay, Connected cars, Mobile platforms, Mirrorlink.	32

3.1.3 Study selection

After the initial search, inclusion and exclusion criteria were derived to filter the results. Firstly, the data extraction included only articles in English. Furthermore, the inclusion criteria was that the articles should propose infotainment platform Automotive Grade Linux and not just mentioning AGL as a possible application area. In addition to the above criteria, additional articles containing the combination of AGL with connectivity, cloud, OTA update, and content streaming were searched.

The inclusion criteria that need to be fulfilled for the mapping study are that the title or abstract mentions that the paper explicitly uses an AGL platform to implement automotive infotainment services. Apart from the papers in English, the exclusion criteria are that the papers just mention AGL but use different platforms to implement automotive services.

All of the articles found were filtered based on the above mentioned criterion. The filtering was based solely on the papers' abstract. No sources were excluded based on the type of study, although the sources after filtering only contained scientific journals and conference papers.

Once the platform was finalized, the researchers carried out the search across all databases using the keywords mentioned above. The filtering resulted in 88 articles for AGL, with 69 articles discarded based on the exclusion criteria mentioned above. Out of the 19 relevant papers, 6 papers were considered a valuable addition to this thesis. These 6 papers provided us the knowledge on the communication between different layers in the AGL architecture, the various API services AGL uses for improving infotainment systems. It helped us to understand why the connected car concept is moving towards open source. It provided knowledge on how different V2C scenarios are implemented in the architecture. It helped us to understand that AGL can be used in combination with other platforms. Finally, it helped us to understand the advantages of using AGL, which is in line with the advantages of an open-source platform. The 19 articles which were discarded did not provide any information on AGL; instead, the papers used different platforms to implement different applications other than infotainment systems. The filtering result for Apple CarPlay and Android Auto resulted in 32 papers, which were discarded as they mention the terms but do not implement Apple CarPlay and Android Auto. The summary of all the papers found across all databases is placed in the GitHub [84]

3.2 Summary of relevant papers

This section summarizes the previous related papers which were found during the systematic mapping study.

Arcangelo Castiglione et al. [71] investigate symmetric encryption algorithms, particularly lightweight ciphers, to secure CAN-level message exchange, which is a reliable solution on hardware-constrained devices. They use the API service of Automotive Grade Linux (AGL), which allows CAN-bus interaction. They propose an approach by implementing appropriate modifications to the AGL system module, which helps in sending messages on the CAN bus. They use Automotive Grade Linux in the experimental phase to check the effectiveness of the proposed solution. They modify the AGL module, which helps in reading and writing the CAN bus message. They use AGL because it contains APIs at all levels from CAN to infotainment, which helps to improve entertainment and safety on board.

In the book, Mobility in a Globalised World 2018, the chapter on "Why Open Source is Driving the Future Connected Vehicle" [70] highlights the importance of Automotive Grade Linux over other platforms like Apertis, Ubuntu Core, SuSe Embedded, Legato, QNX, and Android Auto. They also highlight the major features of AGL like platform features, platform runtime, application runtime, application development, and SDK, app store, licensing, and developer community, which other platforms do not provide. The chapter talks about a possible solution Eclipse kuksa, which is built on AGL and features containing different technologies as services to meet connected vehicle demands.

Mustafa et al.[81] paper discusses the benefits of the embedded Linux platform's challenges, which provides a basis for new improvements. The author discusses aspects that are benefited from the yocto project, which is based on the Linux platform Automotive Grade Linux (AGL). The paper mentions different aspects like architecture, software, and community support, which support people to create features quickly and effectively. This is one of the benefits which is available for the platform. The other benefits include small footprint, platform independence, portability, Manageability and Separation of Concerns, Genericness. The paper concludes by mentioning that the open-source platform is inevitable, which will be more reliable.

In the first virtual All Members Meeting Summit, hosted by the Automotive Grade Linux (AGL) at the Linux Foundation [82]. They mention how V2C (Vehicle to cloud) scenarios can be enabled by connecting AGL to the cloud. In the meeting, they discuss different V2C scenarios like data logging, GPS tracking, remote control, OTA updates, and remote diagnostics and how these scenarios can be enabled independently. They suggest that by integrating cloud connectivity inside the AGL stack, different V2C scenarios can be implemented. They continue by suggesting that Cloud connectivity can be achieved using a set of connectors to different cloud platforms such as Azure, AWS, google cloud, and Bosch IoT cloud. They conclude that each connector acts as a proxy between the AGL ecosystem and cloud platform on the other data where data is transferred to the cloud using a cloud hub.

Stefan [68] writes about how Adaptive AUTOSAR can be used in combination with AGL for updates. Adaptive AUTOSAR provides the software platform to connect the vehicle's ECUs to outside road services. However, to show the significance of adaptive AUTOSAR, a GUI is needed, which is not provided by adaptive AUTOSAR. In contrast, AGL provides a useful GUI implementation that is easy to implement. Thus a combination of adaptive AUTOSAR and AGL is beneficial. The new concept of adaptive AUTOSAR will be introduced, and recent design architectures are outlined. Further, the combination of adaptive AUTOSAR and AGL is motivated to design a proof-of-concept (POC) that is efficient for further performance evaluation of each software platform's software components. In this study, a combination of adaptive AUTOSAR and AGL is proposed to realize a prototype of a connected car software platform.

Sivakumar et al.[69] highlights the characteristics of AGL and how the platform helps automakers for rapid innovation and reduce time-to-market for new products. They then introduce the two automotive gateways with a short discussion regarding constructing a comprehensive in-vehicle communication system with completely different networks and automotive gateways. Thereafter they propose an AGL system where they discuss the technology behind AGL and the software architecture. Furthermore, they discuss the system implementation required to implement an infotainment system using AGL and discuss the hardware setup required to run the AGL image. Their article is similar to this thesis; however, their aim is to show how AGL reduces the lifecycle costs of the vehicle's infotainment system. Whereas this thesis focuses on creating a reference architecture for streaming low latency real-time high-quality content streaming.

4
Methods

In this section, the methods used to conduct the research and achieve the research goal is described. Firstly, the research methodology and outcome in each iteration cycle is explained. Following that, data collection being carried out phase wise is explained. The section ends with the explanation of methods used for each research question.

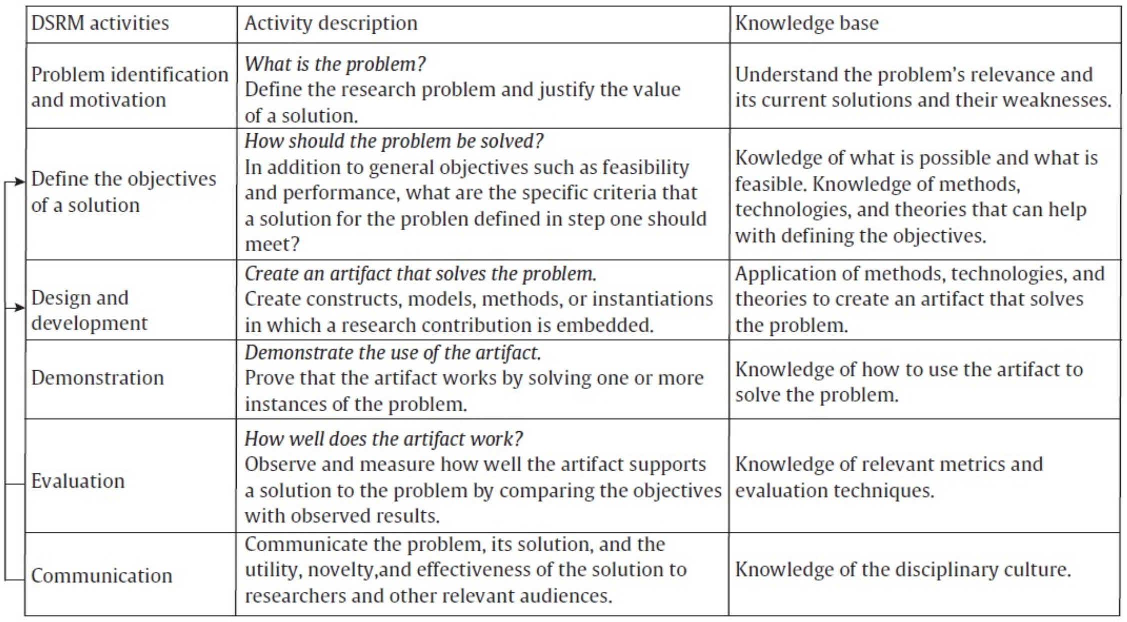
4.1 Research Methodology

Design Science Research Methodology (DSRM) is an outcome-based information technology research methodology, which offers specific guidelines for evaluation and iteration within research projects.[18][19][20]. In this method, the academic research objective is more practical, which suits our thesis topic very well. DSRM was mainly created with three objectives in my mind by Peffers et al. (2008)[20]. They were "(1) provide a nominal process for the conduct of DS research, (2) build upon prior literature about DS in IS and reference disciplines, and (3) provide researchers with a mental model or template for a structure for research outputs."

The primary focus of DSRM is directed towards improving the production, presentation, and assessment of design science research yet being consistent with principles and specifications of design science research created in previous research studies such as Hevner et al., 2004[22], Nunamaker et al.[21], 1991.

Peffers et al. (2008)[20] presented DSRM with a sequence of six different activities: (1) problem identification and motivation, (2) definition of the objectives of a solution, (3) design and development, (4) demonstration, (5) evaluation, and (6) communication. The below table shows the DSRM presented in Peffers et al. (2008) by Guido L. Geerts [19].
Based on the table Figure 4.1, the researchers adapted the format of DSRM for the thesis work as represented in Figure 4.2.

The researchers carried out the thesis using DSRM methodology in an iterative phased manner, as seen in Figure 4.2. The researchers carried out a three iteration cycle. As described in the earlier section, the researcher's problem statement was to develop and evaluate a reference architecture to support near real-time, low-latency, high data volume content streaming from cloud to In-Vehicle Infotainment. The phenomenon being investigated is the functional and non-functional attributes re-



DSRM activities	Activity description	Knowledge base
Problem identification and motivation	*What is the problem?* Define the research problem and justify the value of a solution.	Understand the problem's relevance and its current solutions and their weaknesses.
Define the objectives of a solution	*How should the problem be solved?* In addition to general objectives such as feasibility and performance, what are the specific criteria that a solution for the problen defined in step one should meet?	Kowledge of what is possible and what is feasible. Knowledge of methods, technologies, and theories that can help with defining the objectives.
Design and development	*Create an artifact that solves the problem.* Create constructs, models, methods, or instantiations in which a research contribution is embedded.	Application of methods, technologies, and theories to create an artifact that solves the problem.
Demonstration	*Demonstrate the use of the artifact.* Prove that the artifact works by solving one or more instances of the problem.	Knowledge of how to use the artifact to solve the problem.
Evaluation	*How well does the artifact work?* Observe and measure how well the artifact supports a solution to the problem by comparing the objectives with observed results.	Knowledge of relevant metrics and evaluation techniques.
Communication	Communicate the problem, its solution, and the utility, novelty,and effectiveness of the solution to researchers and other relevant audiences.	Knowledge of the disciplinary culture.

Figure 4.1: Design science research methodology Table as per Peffers et al.

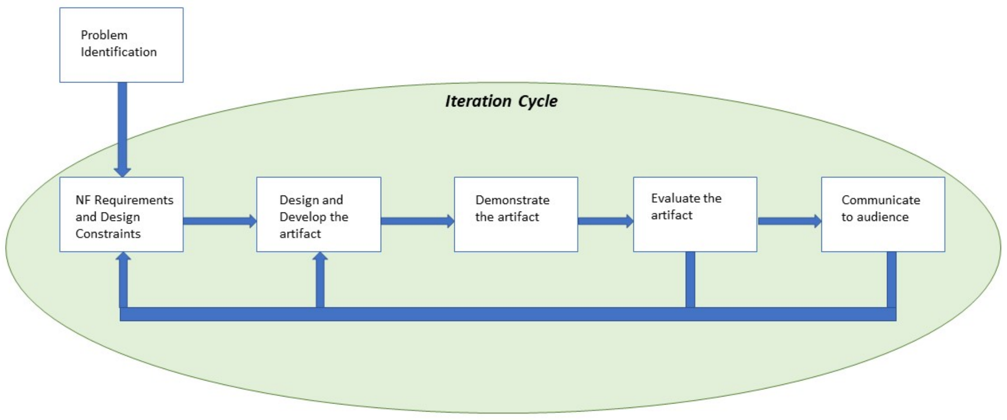


Figure 4.2: Design science research methodology - DSRM [20]

quired to be considered to develop a reference architecture for the cloud to In-vehicle infotainment system. Following that, method to evaluate the reference architecture is looked into. The approach to check how can the AGL reference architecture be extended to support near real-time, low-latency, high data volume content streaming is looked into.

4.1.1 First Iteration cycle

In the first iteration cycle, the researcher's objective was data collection. They need to gather data that would help them understand the current problem and increase the knowledge base to move towards the solution.

Researchers conducted an extensive study of related documents to gather related data about the existing systems. It helped researchers gain necessary background knowledge and possible issues that might arise in architecture development. The document study also helped them to identify current feature support in AGL and upcoming features in its roadmap. This also helped researchers to identify the scope for improvement in terms of streaming content from the cloud.

Researchers carried out a semi-structured interview to acquire the necessary data about existing systems and how the new system should look. The flexible nature of the semi-structured interview was the reason the researchers chose to go with it. It aided in eliciting functional and quality requirements.

Researchers conducted a brainstorming session to identify the possible problems about existing solutions and difficulties they would face in developing the new solution.

The researchers shared the gathered data with supervisors. They took the feedback and suggestions from them at regular intervals. At the end of the first iteration, researchers had gathered reasonable data to begin the work of creating the architecture.

4.1.2 Second Iteration cycle

In the second iteration cycle, the researchers started building the reference architecture. Based on the data collected in the previous cycle, the architecture was created. The researchers created different versions of the architecture, with each version being built on the previous version.

The researchers created various sequence diagrams as seen in Figure 5.3, Figure 5.4 and Figure 5.5 to understand the flow of data in the architecture. The sequence diagrams helped the researchers to identify potential bottlenecks in the architecture. It helped researchers to fine-tune their architecture. Researchers started to develop a prototype to support the reference architecture created using an open-source framework in c++ language.

At the end of the second iteration, the researchers had created the reference architecture.

4.1.3 Third Iteration cycle

In the third iteration cycle, the reference architecture built in the previous cycle was evaluated using interviews. The interviews were conducted among individuals who had good knowledge of AGL and reference architectures.

Researchers completed the prototype for reference architecture. At the end of the third iteration cycle, researchers had gathered the interview result, analyzed it, and drew conclusions towards the research goal.

4.2 Data Collection

The main objective of data collection is to help answer the research questions. This study's data collection is a combination of documents study, interviews, and brainstorming categorized as qualitative research. The data collection was conducted in three phases. Phase 1 was conducted to gather relevant data about existing systems and to understand its current limitations. Phase 2 was conducted to elicit functional and non-functional requirements, which were needed to develop reference architecture. Phase 3 was conducted to gather evaluation results of reference architecture developed.

In phase 1, researchers started data collection by performing a document study. This method helped the researchers to collect critical data about the rules and the relevance of the existing systems. Analysis and study of the data showcased a brief insight into the possible path in this study and how to proceed.

In phase 2, the outcome of findings in phase 1 was used to formulate and structure the semi-structured interviews and brainstorming approaches to elicit functional and non-functional attributes needed to design and develop the reference architecture. Data collected from this phase helped in moving one step forward towards finding the answer to research questions.

In phase 3, semi-structured interviews were conducted to obtain evaluation result data of the architecture. The data from all the above steps were combined to answer the research questions.

The data collected from phase 1 and 2 laid the foundation for developing the key components of cloud architecture for streaming data to the in-vehicle infotainment system. The Figure 4.3 presents methods used in data collection approach.

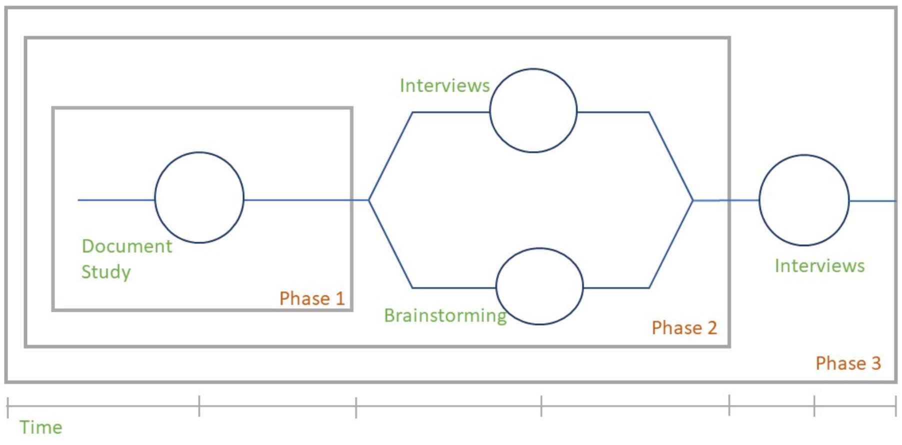


Figure 4.3: Representation of methods used in data collection approach

4.3 Research Question 1

The researchers used document study, brainstorming, and interviews methods for RQ1. Document study was done in common to answer both RQ1 and RQ3. Document study will be explained under subsection 4.5.1.

4.3.1 Brainstorming

Brainstorming is a technique where ideas are shared openly [23] [24]. The researchers learned about this technique during their Master's course - Requirement Engineering. The technique was chosen as the researchers wanted to debate the possible requirements for the architecture quickly and efficiently.

The brainstorming was done as a formal session, where the researchers had booked a meeting room and a specific time duration was set. The agenda was set to elicit the requirements (functional and non-functional) for developing the reference architecture and to identify possible bottlenecks. The researchers performed two sessions of brainstorming, with each session lasting approximately 120 minutes. The outcome of the session was documented. The result will be discussed in the result section.

The two researchers have previously worked as software engineers in the software industry for 8+ years for various companies before starting the master course. They have worked on various projects during their work tenure and have interacted with many international clients during the course of work. This work background of researchers helped them to use this technique effectively to come up with a broad range of ideas for requirements.

This also provided an opportunity to go in-depth and explore possible bottlenecks that might arise while developing the architecture. The possible constraints were identified, and an approach to solving them was constructed. The innovative thoughts generated were useful and documented.

4.3.2 Interviews

Semi-structured interviews were chosen to gather the existing system's required information and how the new system should look. The researchers selected semi-structured interviews as the data collection method mainly due to their flexible nature. The core of the intended question is kept, and its flexibility to modify based on the interviewees was apt for the type of data researchers were looking for. These were mainly conducted to elicit quality and functional attributes for the In-Vehicle Infotainment domain and cloud streaming domain in general. A set of prepared questions was asked to the interviewees, and follow-up questions were asked based on context. The questions were designed in order to gather information about existing cloud and infotainment systems. It also covered the expected quality and functional attributes that needed to be part of the proposed architecture development. The questions can be found in section A.1. In order to assure the quality of the questions, the questions were reviewed by a supervisor from Aptiv with sound knowledge about the existing In-vehicle Infotainment system and by Chalmers University professor. This was of importance as the questions need to be pertinent, clear, and comprehensive to an appropriate level.

Researchers chose the interview participants who had good work experience in software industry and had adequate knowledge on infotainment systems. Researchers interviewed a total of 12 people (software engineers working in Aptiv). The interview respondents composed of Lead developers/Architects, Senior developers and Junior developers working in Aptiv. All interviews conducted were documented, and the transcript was prepared for analysis by researchers. The consent of the interviewees was taken to share and document the data gathered for thesis purpose. Interview meetings were planned and booked in advance, and they were conducted during office working hours. Most were held in conference rooms.

The interview sessions were planned for the duration of 45 minutes and were informed to the interviewees beforehand. The researchers had kept a buffer of 15 minutes as well. Both the researchers were present during the interview. One of them asked the questions, and the other was taking notes of the response. Interview sessions started by thanking the subjects for their participation, followed by a brief introduction about the purpose of the interview and the topics being covered in the questions. The interview flow and confidentiality were also addressed. The entire interview process was audio recorded with appropriate consent from interviewees. The interview sessions were structured according to the Pyramid model, which begins with specific questions and continues with open-ended ones.[30] At the end of the session, feedback and suggestions from subjects were noted.

After each interview session, the session's summary list based on notes taken was prepared, and the knowledge gained was noted. The audio recording was transferred into a text transcript for future analysis.

4.4 Research Question 2

The researchers used Interviews to evaluate the attributes and architecture developed for research question 2.

4.4.1 Interviews

The researchers decided to use a semi-structured interview to evaluate whether the reference architecture satisfies the functional and non-functional attributes identified in RQ1.

The researchers set up an online zoom meeting for the interviews. The interview consisted of 15 questions, and the estimated time to completing the interview was 20 minutes. The researchers aimed to keep the duration short so that the participants accept to be part of it. Another advantage of having the interview duration shorter is that response quality will be better and will keep the participants more motivated[53]. The researchers have taken great care in keeping the questions short and addressed to the point, keeping in mind that participants might not be willing to spend too much time in the interview.

Researchers constructed two types of questions for the interview: open-ended and closed-ended questions. The questions are can be found in section A.2. The open-ended questions were designed to give the participants a free hand in voicing their opinions. This would give participants more freedom to express their points of view. The closed-ended questions were designed as multiple choices/checkboxes and Likert scale questions. This was designed to retrieve quantifiable data. The researchers have put in their best effort to make the questions very precise, clear cut and explicit. The college supervisor reviewed the questions to improve the quality of the questions. The survey included few demographics questions, followed by general questions related to AGL; then it delved into questions relating to the evaluation of attributes in architecture identified as part of interviews and brainstorming. It also included questions about the understand-ability and usability of new components added as part of a reference architecture to existing AGL architecture. The general suggestions were also asked at the end of the interview.

The selection of participants was made with care, as it had a pre-requisite of good knowledge about AGL architecture. The researchers interviewed a total of 5 people. They chose one working software professional from Aptiv company. Then they approached four IEEE paper authors who have published papers on a topic related to AGL and Infotainment architecture in general, by sending out an email request to be part of the interview. The email consisted of a brief about the thesis and its purpose, followed by the estimated time taken to complete the interview. After the participants accepted the invitation, researchers finalized the list of 5 participants and then sent the details about the architecture and its component beforehand. The researchers set up the zoom meeting based on common time availability, and the zoom meeting invitation was sent. The participants were sent the architecture details approximately one week before the zoom meeting.

In the interview session, the researchers greeted the participants and thanked them for taking time out of their schedule to be part of the interview. The consent of the participants was taken to use their response data. The participants were also informed that their responses and analysis would be available online as part of thesis publication.

The researchers gave a presentation about the problem being addressed in the thesis, its motivation, and the details about the proposed reference architecture and the components. The UML diagrams were presented to illustrate the workflow of the architecture. The presentation lasted close to 5-8 minutes, majorly due to the fact that the participants had read the architecture detail document sent prior to the meeting. Post presentation, the researchers went ahead with the set of interview questions. One of the researchers was handling the part of asking the question, while the other was taking notes of the participant responses. At the end of the interview, the researchers thanked the participants again for their valuable time. After each interview session, the summary list of the session based on notes taken was prepared, and the knowledge gained was noted. The average total interview session was approximately 25 minutes of duration.

4.5 Research Question 3

The researchers conducted a document study to answer research question 3.

4.5.1 Documents Study

Document study was the initial step conducted by researchers to gather related data about the existing system. The initial study began by reading about the existing cloud system architectures, current infotainment system, followed by the architectural styles used by Aptiv company for their infotainment solutions. Then researchers studied Automotive Grade Linux(AGL). They studied the existing feature support by AGL and its future road-map. These documents were carefully studied and analyzed by both the researchers. Few key documents were studied, which gave the researchers useful insight into the existing solutions (AGL and other platforms) and the various reasoning behind selecting the existing solution. The main business logic of the systems was comprehended. This gave us an important understanding and purpose of the current systems. It aided researchers in identifying the gaps and scope for improvement in AGL in terms of content streaming from the cloud. It also helped identify how to evaluate quality attributes of architecture and measure quality attributes of a reference architecture.

All the documents helped the researchers to comprehend the required background knowledge for developing the architecture. Additionally, it also helped researchers to identify possible problem areas in developing the architecture solution. The most important complex areas were identified by researchers and helped them in understanding the whole picture.

5
Results

This section presents the results from interviews, brainstorming and document study and, software architecture developed. These findings are based on systematic experiments involving human subjects. The first section presents the software architecture developed, followed by the sequence diagram for the architecture developed. The second section presents the results for interviews, brainstorming, and document study for research question 1. The third section presents the results for interviews for research question 2. The section ends with document study results for research question 3.

5.1 Architecture

In this section, the reference architecture designed is explained. The architecture was developed by taking AGL as a reference, and the architecture is based on the data which was collected during the data collection phase. According to the data collection and analysis, it can be seen that the quality attributes like availability and performance were the most preferential. To render the system when there is a drop in connection, the architecture utilizes an offline manager, which fetches the cached data, and the response is streamed to the UI, making the system available at all times. The infotainment-cloud state sync manager is used for state synchronicity management. It helps to keep the infotainment system's state in sync with the cloud between online and offline mode.

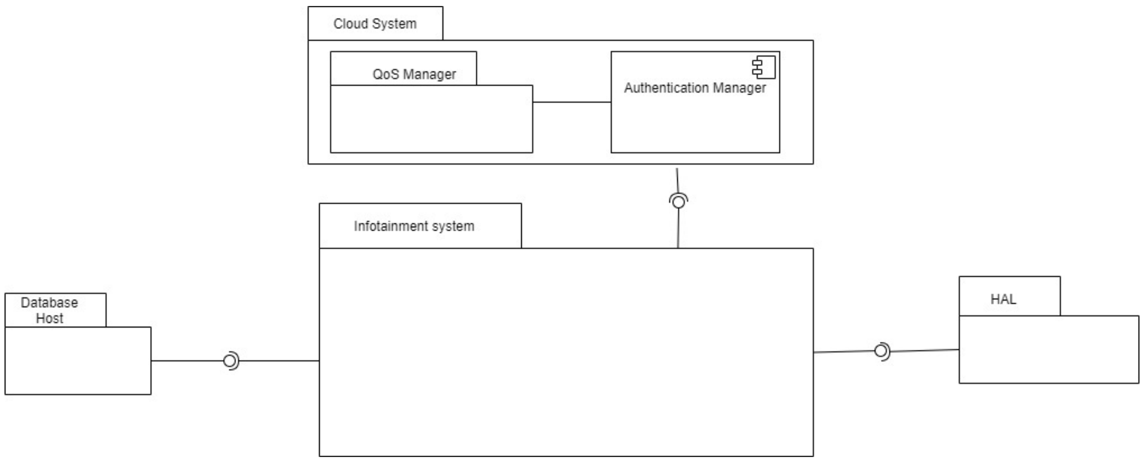


Figure 5.1: Top level component diagram of the cloud-infotainment system

Figure 5.1 represents the top-level diagram of the cloud - infotainment system. The infotainment system interacts with the cloud system to get data for the requests it makes. It also authenticates itself with the cloud system. Database host sends responses for any data request/records made by the offline manager. The infotainment system sends service requests to HAL to perform hardware-related activities. The user manager gets the response from the cloud. It sends the response to the Infotainment UI via the Policy Manager.

Figure 5.2 represents the detailed reference software architecture of the infotainment system. The reference architecture consists of two parts - Cloud system and Infotainment system. In this thesis, we will discuss on Infotainment system part of the architecture and cloud system part is only to provide a reference to picture the entire system as a whole. The infotainment part of reference architecture consists of 9 components. Next section provides the detailed description of each of the components used in the reference architecture developed.

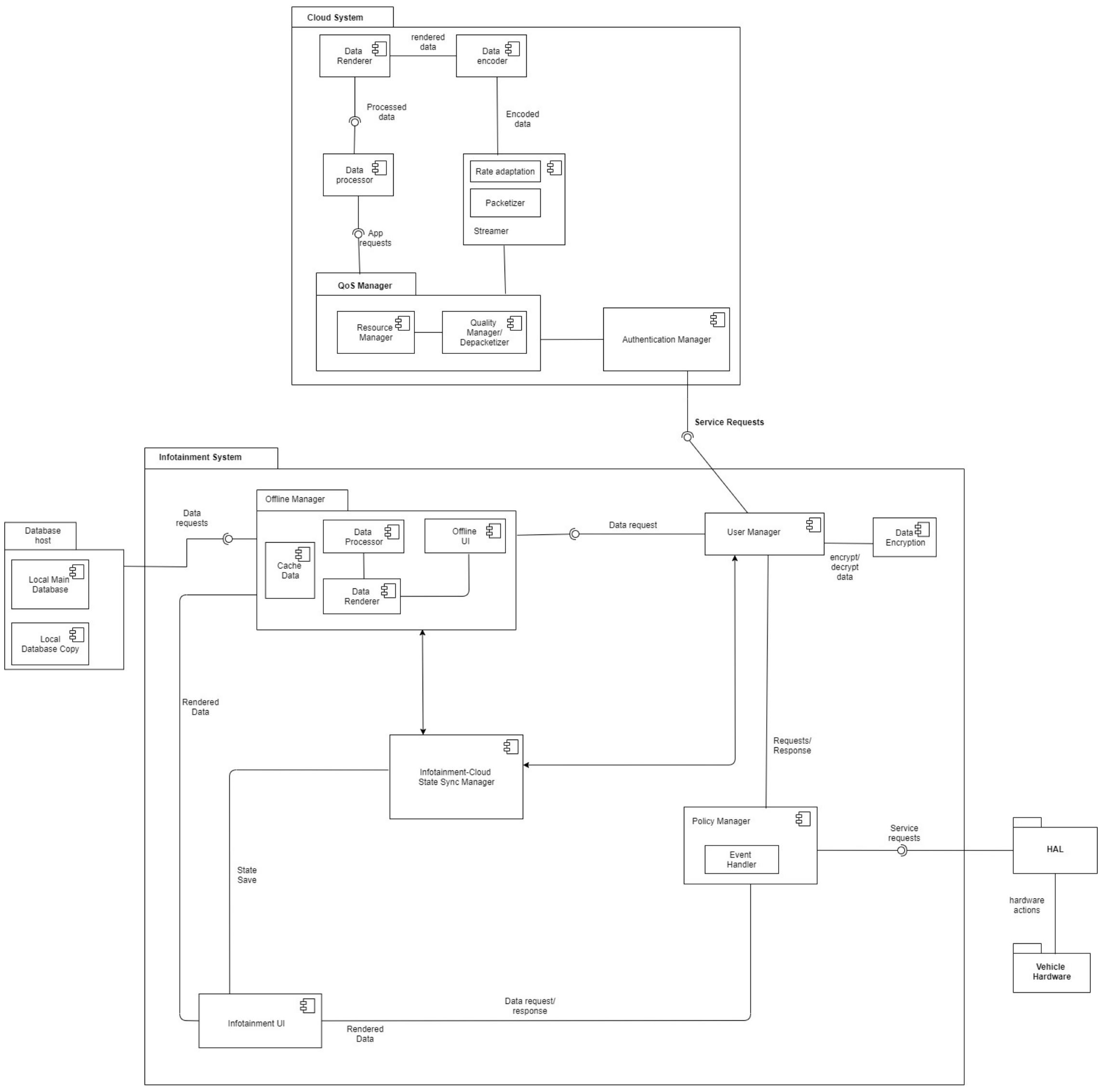


Figure 5.2: Reference Software Architecture

5.1.1 Component Description of the reference software architecture

This section presents the description of the components which are used in the software architecture developed.

1. **Database Host** This component stores the data that needs to be sent when the system is in offline mode. In offline mode, the Offline Manager requests data from the local database to be sent to the UI.

2. **Offline Manager** This component is responsible for processing the data request from the User manager in offline mode. This component stores the cloud's response in cached data, and this data is sent to handle UI requests. When there is no cache data available, the Offline Manager, with the help of

an offline UI and data processor, fetch the data from the local database and send it to the data renderer. Data Renderer renders the data obtained by combining the data from offline UI and data processor.

3. **User Manager** This is an existing component in AGL. In addition to its existing responsibility, we have added the following responsibilities. This component checks for the connection at regular intervals between the infotainment system and the cloud. On a data request from the Policy manager, and if there is a connection available, it sends a request to the cloud. The response from cloud is sent back to the Policy manager. It also sends the data response from the cloud to the Offline Manager. Where there is no internet connection available, it sends the data request to the Offline Manager. When the internet connection is changed from offline to online and, if there is a state synchronization request from the Infotainment-Cloud State Sync manager, then it sends the request to the cloud. The corresponding response from the cloud is sent back to the Infotainment-Cloud State Sync manager. The communication between the User manager and the cloud is encrypted.

4. **Policy Manager** This is an existing component in AGL. This component is mainly responsible for making decisions regarding the data request. When there is a data request, the Decision Manager decides whether the requests need to be sent to the cloud for the response or whether it can be handled locally.

5. **Infotainment-Cloud State Sync Manager** The Infotainment-Cloud State Sync Manager stores the latest state of the Infotainment UI at the time of disconnection. This component on regular intervals will check with the User Manager that if there is a connection established to the cloud. Once reconnection is established, the latest state data of UI will be sent by Infotainment-Cloud State Sync Manager to the cloud via User Manager. The cloud resolves the state and sends the updated state. The Infotainment-Cloud State Sync Manager receives the updated state, updates the current state, and sends it to the Offline Manager to update the UI.

6. **HAL** Hardware Abstraction Layer takes care of communicating with vehicle hardware to send and receive data.

7. **Vehicle Hardware** The vehicle hardware to which data is sent/received form HAL.

8. **Event Handler** This component processes the inputs from the UI and handles the request to be sent to the cloud and hardware. This component helps to achieve performance, where it will limit the number of requests.

9. **Offline UI** This component is responsible for generating a UI in case of connection drop and the generated UI is sent to the data renderer, which in turn

streams it to the infotainment UI.

5.1.2 Sequence diagrams

In this section, three sequence diagrams for the software architecture are presented. They depict the component interaction in three scenarios.

Figure 5.3 represents the sequence diagram when there is internet connection between Cloud and infotainment system. The Infotainment UI sends out the request to Policy Manager. The Policy Manager decides whether the request needs to be sent to cloud or HAL. For requests to be sent to cloud, Policy Manager sends the request to User Manager. User Manager checks for internet connection with cloud. On successful connection response, User Manager sends the request to the cloud. The Cloud system sends data response to User Manager, which is sent to Infotainment UI via Policy Manager.

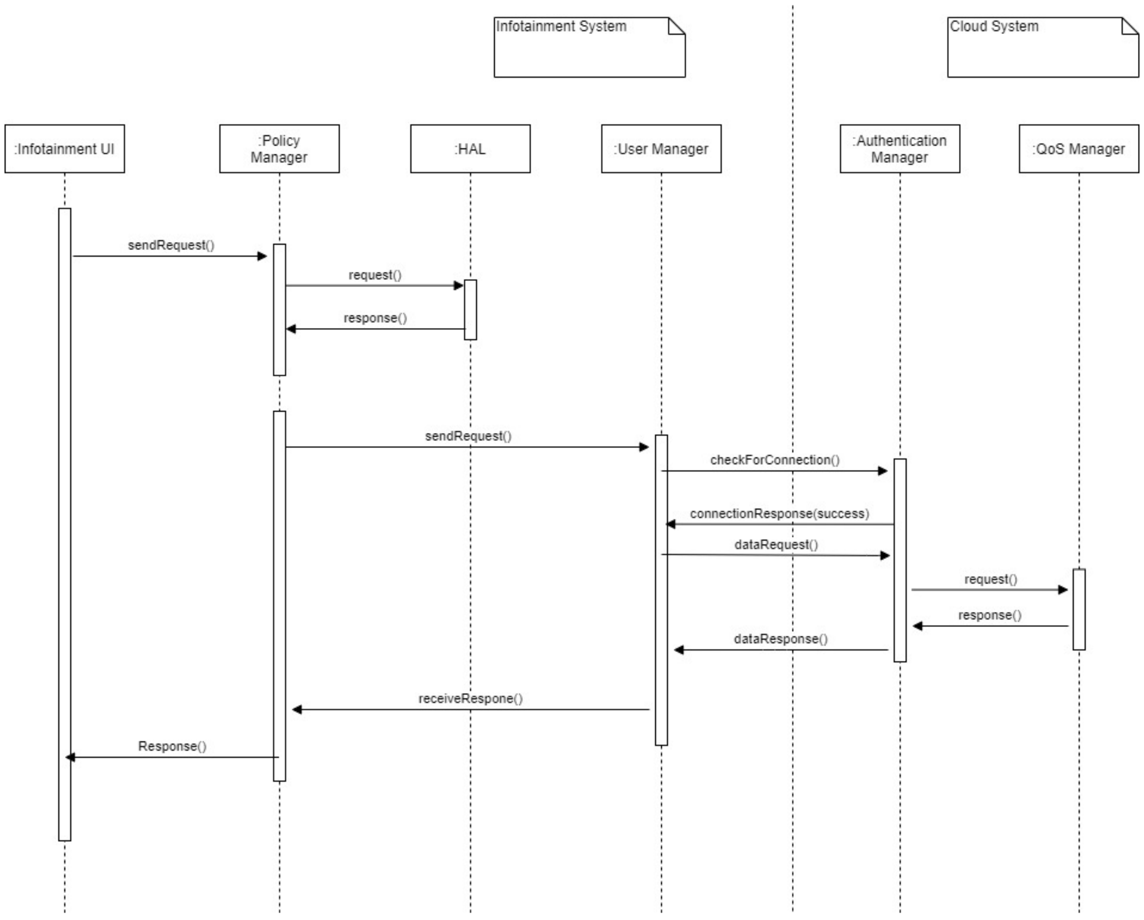


Figure 5.3: Sequence Diagram with internet connection

Figure 5.4 represents the sequence diagram when there is no internet connection between Cloud and infotainment system. For the request that needs to be sent to cloud, the Policy Manager sends the request from Infotainment UI to User Manager. User Manager checks for internet connection with cloud. On unsuccessful connection response, User Manager sends the request to Offline Manager. Offline Manager checks for cache data, processes the request and sends data response to the Infotainment UI.

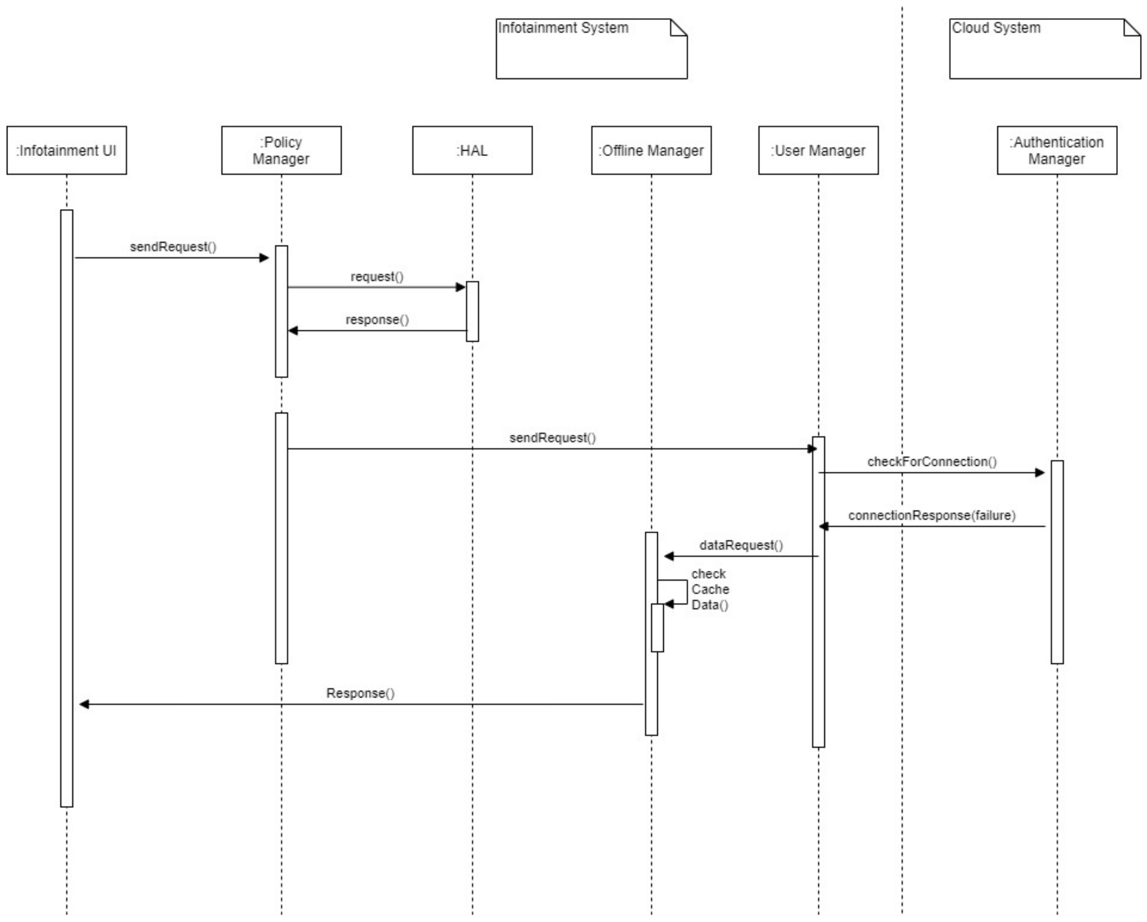


Figure 5.4: Sequence Diagram without internet connection

Figure 5.5 represents the sequence diagram for state synchronization scenario between cloud and infotainment system. When there is no internet connection between cloud and infotainment system (offline mode), Offline Manager sends the response to Infotainment UI. Infotainment-Cloud State Sync Manager component gets the latest state of the UI and periodically checks with User Manager regarding internet connection status. On re-establishment of internet connection between infotainment and cloud, the Infotainment-cloud state sync manager sends the latest state of the UI to the cloud via User Manager. The cloud synchronises the states and sends the updated state to the Infotainment-cloud state sync manager via User Manager. The Infotainment-cloud state sync manager updates its current state and sends the updated state to Offline Manager, which updates the infotainment UI state to latest state from cloud.

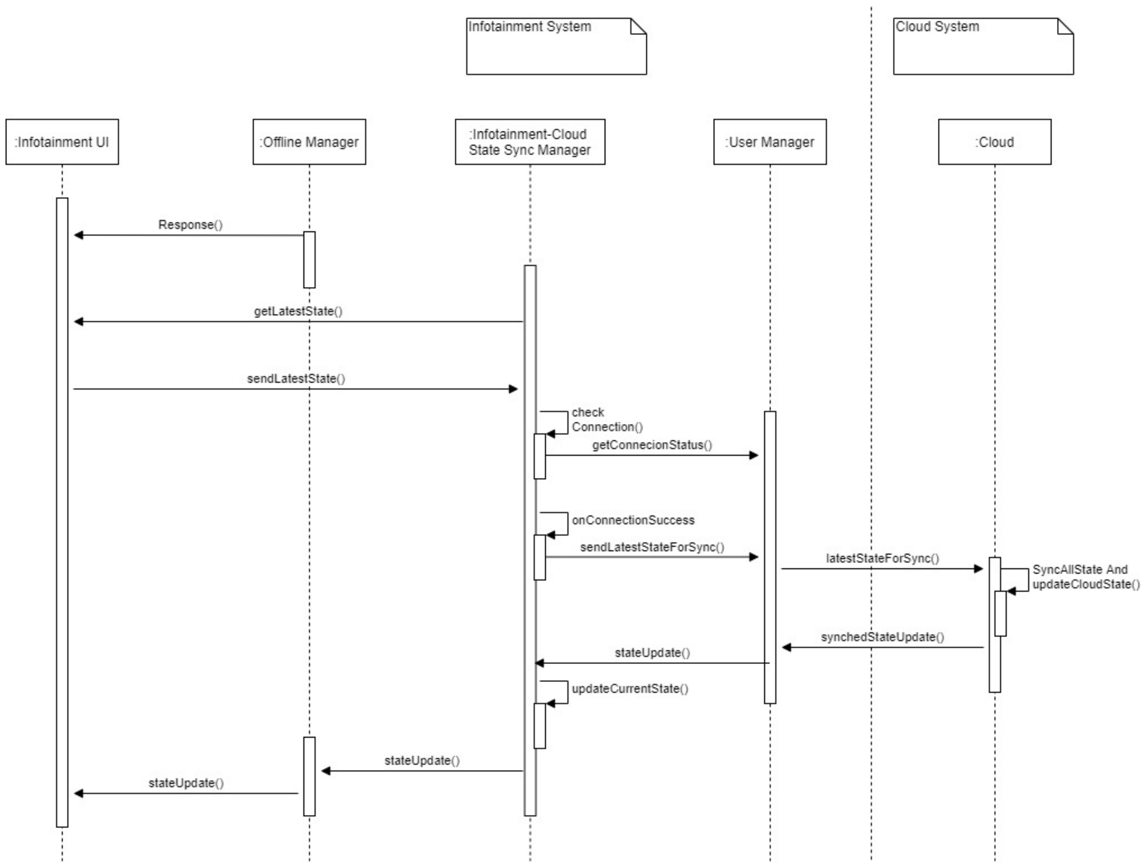


Figure 5.5: Sequence Diagram for Synchronization scenario between cloud and infotainment system

5.2 Findings for Research Questions

To be able to answer the research questions, several steps had to be made beforehand. Related literature had to be analyzed. Interviews had to be conducted to elicit the functional and non-functional requirements required to develop the architecture. Finally, the developed architecture had to be evaluated using semi-structured interviews. After these steps were completed, results could be generated to answer the research questions.

The Figure 5.6 represents the different times at which the data collection was performed and analyzed. Data from the document study was collected and analyzed in phase 1. The data from brainstorming and semi-structured interviews were collected and analyzed in phase 2. The data from semi-structured interview to evaluate the architecture was collected and analyzed in phase 3. .

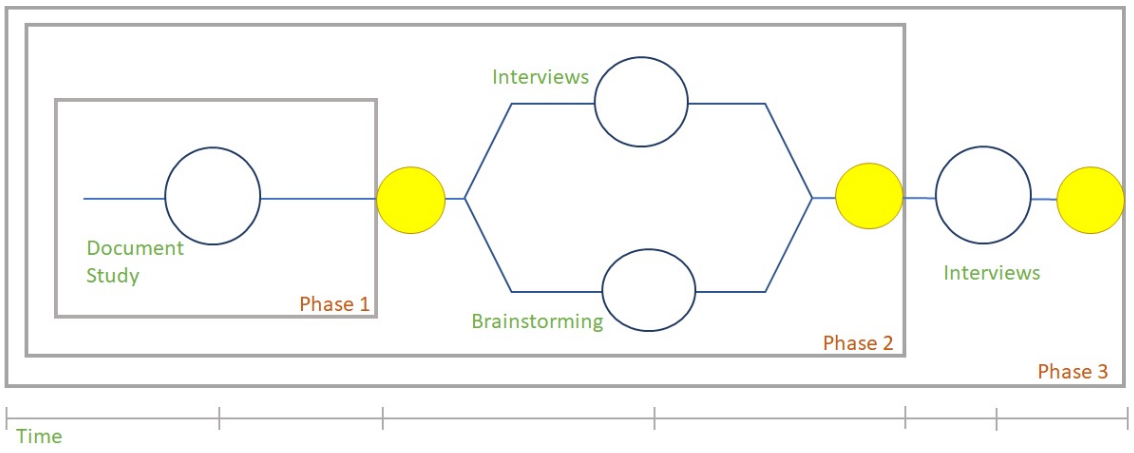


Figure 5.6: Representation of methods used in data collection approach

5.2.1 Research Question 1

5.2.1.1 Findings from Document study

In this study, qualitative content analysis was used for data analysis of the documents study. Researchers recognized the critical concepts and patterns by analyzing the documents.

The researchers thoroughly analyzed the initial set of documents that were gathered during the first phase of data collection. The data analysis of the document study allowed researchers to understand the core concepts with clear sight and in-depth perception. This approach helped the researchers to concentrate better on the most critical areas and study them in full depth.

Researchers obtained a good understanding of the existing cloud-based and game-based architectures. It helped them to acquire knowledge on key components essential for architecture for cloud streaming of infotainment for an automobile. The different available architecture styles were thoroughly analyzed by researchers, which helped them to grasp the general overview of available and popular architectural styles. Researchers analyzed top-rated cloud-based applications like Google Stadia and Spotify. Analyzing these applications helped the researchers understand the architectural styles used by them. The similarities and dissimilarities between the existing architecture and the one proposed to be developed were analyzed. It helped the researchers to focus better on specific required areas.

Based on the knowledge obtained in the document study, researchers identified the need for a reference architecture for streaming the IVI from the cloud. The scope of the architecture is large for the problem statement. Hence, with the help of supervisors, researchers reduced the scope to the infotainment system part only. The cloud part is not being discussed in this thesis. This helped researchers to focus with better clarity to work on the architecture.

5.2.1.2 Findings from Interviews and Brainstorming

In this study, the data set obtained from interviews and brainstorming was analyzed. In a brainstorming session, the researchers shared their ideas about the functional and non-functional requirements for developing the reference architecture. During the session, researchers started to discuss the most important attributes present in the current infotainment system. This initial discussion went on for about 60 minutes. During this short session, the researchers came up with many quality requirements. They noted down these requirements. Then researchers started to discuss the attributes that will be important when developing a cloud-based infotainment solution. They made a note of these requirements. This short session went to about 60 minutes. Researchers then took a short break and started with the next session of 120 minutes. In the first half of this session, they discussed in detail the pros and cons of each requirement they made a note of in the previous session. The various trade-offs associated with the requirement were debated. This session went for about 90 minutes. Researchers then discussed various functional requirements that might be necessary for the cloud-based infotainment system. This discussion went for about 30 minutes, and the ideas were noted down. At the end of the two sessions conducted, researchers agreed that Availability and Security would be the top two non-functional requirements that would be most necessary and should not be compromised in the architecture. For functional requirements, they agreed that the navigation feature should be an important functional feature that should be needed for the Infotainment system, as most of today's users use and expect navigation to be part of the infotainment. Researchers were also of the opinion that climate control (AC temperature) should also be another important functional feature. This is because having a climate control feature has become a standard norm in most today's cars, and users will more often use this feature when driving. This feature also helps the users to maintain the desired temperature inside the car, which will make them relaxed while driving, as outside temperature varies and cannot be controlled.

The interview data set was categorized based on the type of question by researchers. Various topics were covered in the questions. There were few personal questions asked, like the interviewees' job role, their overall work experience, and their work experience in the infotainment domain. Then there were domain-specific questions on cloud infotainment. They were asked about current features in infotainment systems, their quality and functional attributes, and their challenges. The interviewees were asked about cloud-based architecture, challenges associated with it, and the necessary top attributes. At the end of the interview, researchers asked for general suggestions apart from the questions asked. The set of questions can be found in section A.1

There was a good mixture of the composition of the interviewees. Based on their job designation/role, about 50% of the interviewees were Lead developers/Architect, 30% were senior developers, and 20% were junior developers in the software industry. Based on the tenure of work experience in the software industry, about 70% of interviewees had 8+ years of experience, 20% had 5-8 years of experience, and 10% had less than five years experience working in the software industry. Based on working in the Infotainment domain, about 60% had 5+ years experience, 20% had 2-5 years experience, and 20% had 0-2 years experience in the infotainment domain. A graph was created to illustrate the data obtained in a human-readable format. The Figure 5.7 depicts the quality attributes desired for the proposed architecture development.

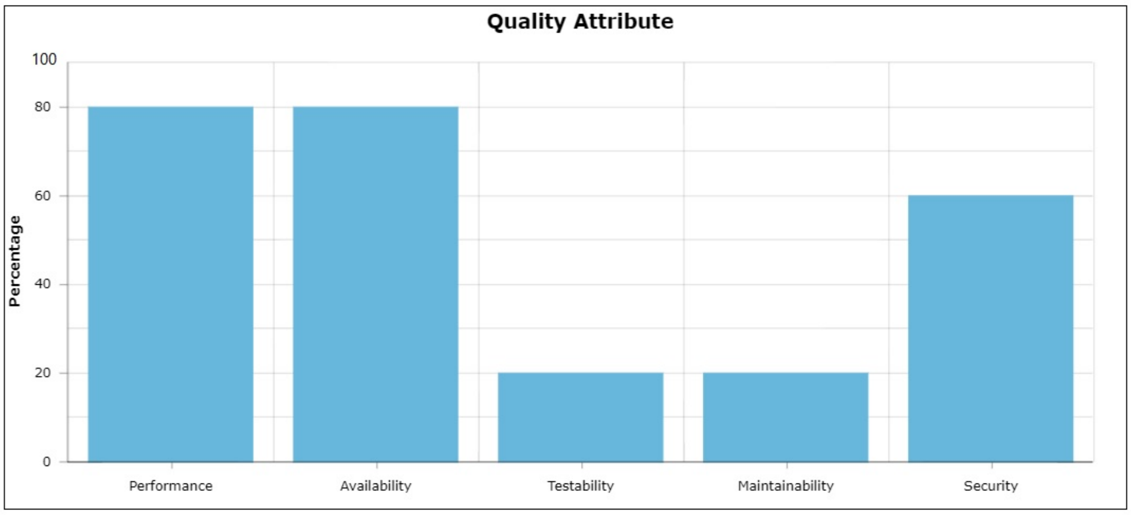


Figure 5.7: Data gathered from interview for Quality Attribute

The Figure 5.8 depicts the functional attributes desired for the proposed architecture development.

As represented in Figure 5.7, About 80% of the interviews were of the opinion that Performance and Availability was an important quality attribute, 60% think Security was an important quality attribute, and 20% think test-ability and maintainability was an important attribute. Based on interview data, researchers analyzed that Performance, Security, and Availability are the top most important quality attribute

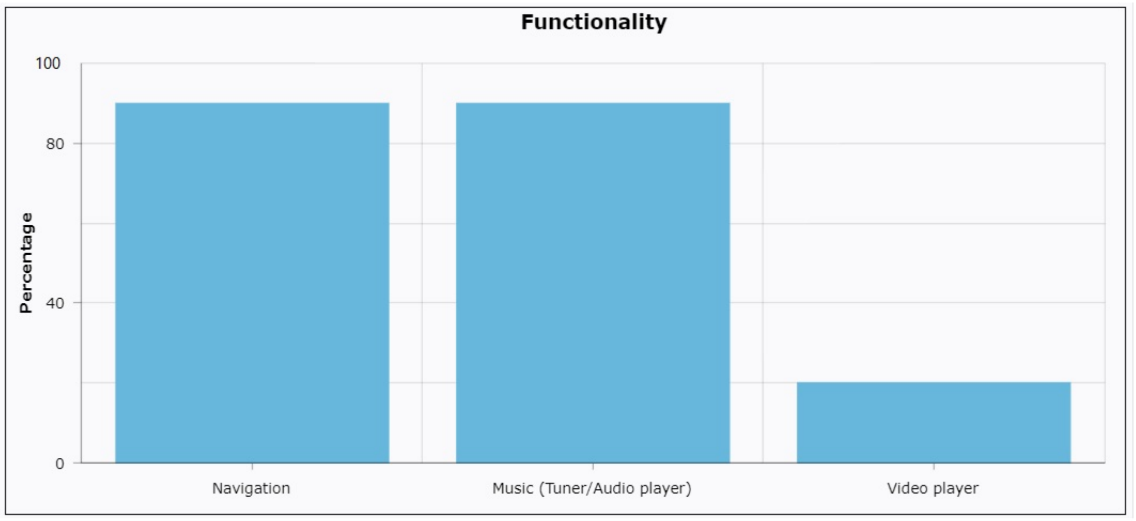


Figure 5.8: Data gathered from interview for Functionality Attribute

that is expected of the architecture. The majority of interviewee subjects emphasized these three quality attributes for the system we planned to build.

As represented in Figure 5.8, about 90% of interviewees believe that Navigation and Music (Tuner/Audio Player) is the most important functionality of the infotainment system. 20% of interviewees think video player functionality is important. Based on data, researchers analyzed that Navigation and Music (Tuner/Audio Player) to be the most important functionality that has to be supported in IVI.

Analysis of the interview data has helped the researchers concentrate on the important attributes for the development of architecture. It enabled them to make wise trade-offs and apply appropriate tactics and styles in developing the architecture.

5.2.2 Research Question 2

A semi-structured interview was used for Research question 2. The set of questions can be found in section A.2. There were a total of 5 participants who took part in the interview. 4 participants were men, and one was a woman participant. 20% of the participants were from Sweden, 40% of them were from other parts of Europe and 40% were from India.

Figure 5.9 depicts respondent's various roles and responsibilities as per question 1 in interview questions. It can be clearly observed that the Software Architects and Senior Developer/QA were the highest respondents with 50%. Students (researcher) and Product owners compromised of 25% each.

Figure 5.10 illustrates the knowledge or understanding about AGL as per question 2. It shows that 50% of the respondents had read white papers about AGL and have used AGL in prototype projects. 25% of the respondents had been involved in product development based on AGL, and the remaining 25% had just heard about

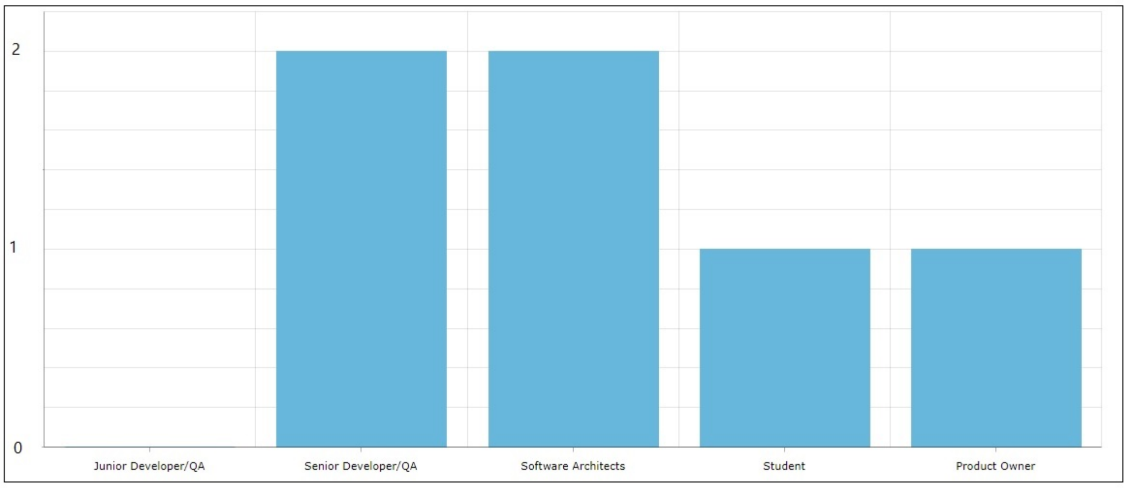


Figure 5.9: Role/responsibilities of respondents

AGL. There were no respondents who had not heard about AGL.

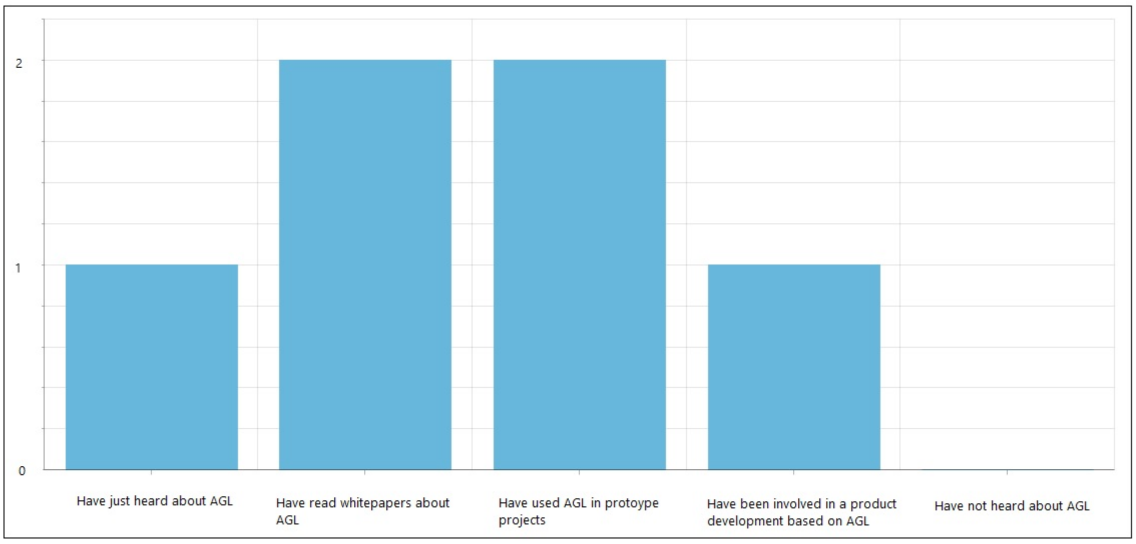


Figure 5.10: Knowledge/ understanding of AGL

In the next question, the respondents were asked to rate their level of understanding of the roles/responsibilities of the architecture components based on the description provided prior to the interview.

Figure 5.11 depicts their responses to the level of understanding. It clearly shows that for the Offline Manager component, 60% of the respondents felt they understood the design for the Offline Manager component, but would not feel comfortable explaining it to others. 40% of the respondents felt that they could understand the design and would feel comfortable to explain it to others, but could not make changes to adopt it our needs. For Infotainment-cloud state Sync Manager, 40% responded that they could understand the design and feel comfortable explaining to others, but could not make changes to adopt it our needs. 60% responded that they understood the design but would not feel comfortable to explain to others. For User Manager and Policy Manager, 60% responded that they could understand the design and would feel comfortable to explain it to others, and could make changes to adapt it our needs, whereas 20% responded that they understood the design, but would not feel comfortable to explain it to others and 20% responded that they could understand the design and would feel comfortable to explain it to others, but could not make changes to adopt it our needs.

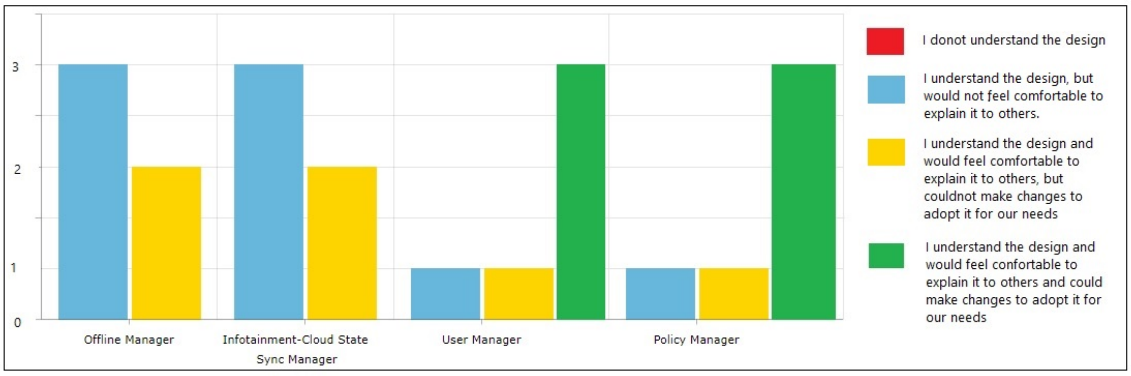


Figure 5.11: Level of understanding of reference architecture components

In question 4, the respondents were asked to rate the possible end-to-end data latency for UI interaction in our infotainment system. Figure 5.12 depicts that 60% of the respondents were of the opinion that the end-to-end latency would be between 10-50 milliseconds(ms). 20% responded that latency would be between 50-100ms and the remaining 20% responded with 100-150ms.

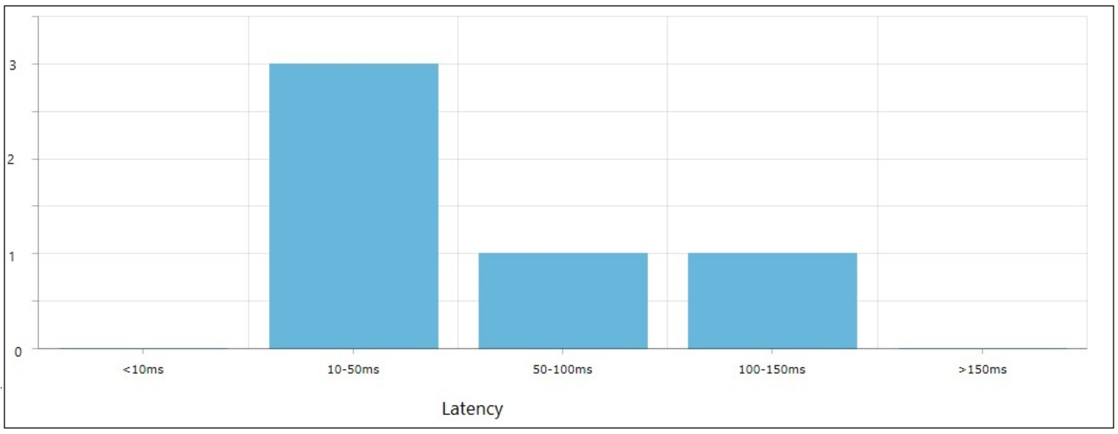


Figure 5.12: End-to-end latency for UI interaction in our infotainment system

5.2.2.1 Infotainment cloud state sync manager component

The next set of questions were based on the Infotainment cloud state sync manager component. In the interview, the respondents were explained about the role of the component, which is to synchronize the state between cloud and infotainment system.

Question 5 asked about the respondents opinion about the role of this component. One of the respondents, the senior Architect, indicated that this component achieved state sync between online and offline systems and thought that it was a good idea to reduce the transfer of the state at a given point in time.

Few respondents were of the opinion that the component is essential and very important to achieve synchronicity of the state between online mode (internet connection) and offline mode(no internet connection). The product owner responded that the need for state synchronization management is high in many products that he handles in his software company. The respondent indicated that the state sync manager sends the latest state data of the UI to the cloud via User Manager and thought it would play a major role in achieving state synchronicity in the case of internet re connection.

One of the respondents was of the opinion that it depends on implementation and did not want to comment on this question.

The respondents were asked about the possible latency addition due to the Sync manager component as per question 6. Figure 5.13 illustrates that 60% of the respondents were of the opinion that the latency added will be very negligible and less than one milliseconds(ms). The rest 40% stated that latency might be between 1-5 milliseconds.

The respondents were asked the basis of choice of their responses. They stated that having low latency in the Infotainment-Cloud State Sync Manager component will provide a better user experience for the infotainment system and thought it should have low latency. They felt this component would play a crucial part in user experience, which will depend on the implementation of the component. One of the respondent, the researcher, felt this component would play major role in offline mode and indicated the latency will be less. The respondent thought that this component does not appear to interfere with the online part, and it kept the state in sync. The respondent liked the idea of states being separated out.

The respondents were of the general opinion that this component would be a good value addition to the architecture. Following their latency expectation, the respondents were asked about possible drawbacks and how it can be mitigated to achieve better state synchronization as per question 7.

The respondents came up with various answers. They thought about possible drawbacks that this component might offer or face. The software architect indicated that he could not identify any major drawback in the component. He thought that, as the component is an addition to the already existing architecture, the entire system's latency would not be compromised.

Few respondents were curious to understand that when new states are added, how will the component track them. They felt that if tracking is not implemented correctly, the system might face state management issues. One of the respondents raised few concerns, like how the Infotainment-Cloud State Sync Manager component supports the transient between two states and whether it is possible to increase or decrease the component's latency.

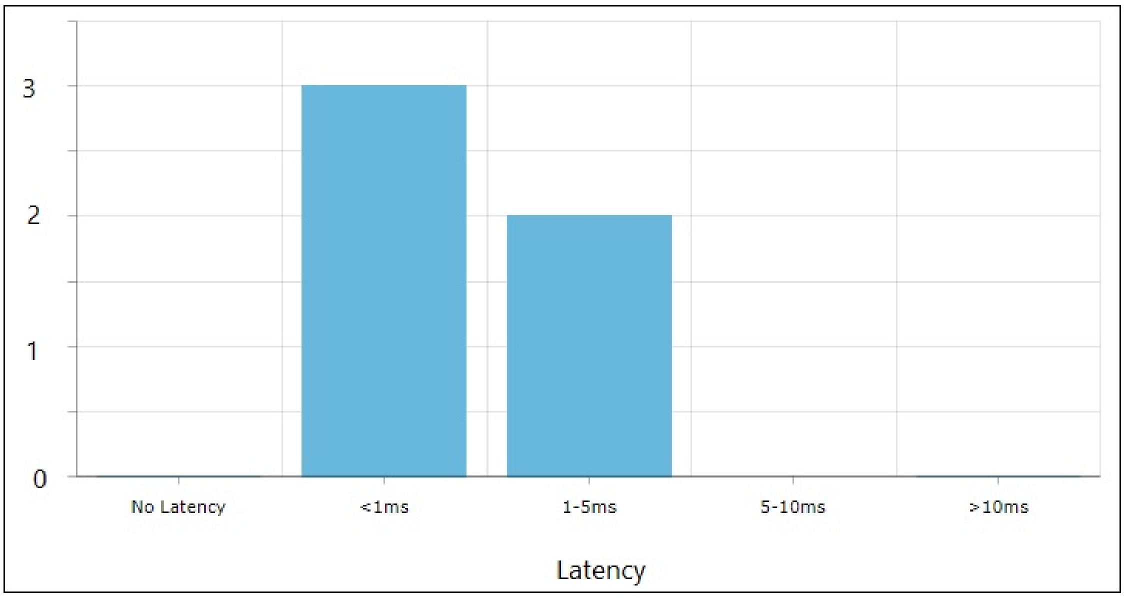


Figure 5.13: Latency added by Infotainment-Cloud State Sync Manager

5.2.2.2 Offline Manager

Following the Infotainment-Cloud State Sync Manager component, the next set of questions were based on the Offline Manager component. The respondents were explained about the Offline Manager component and its role in the reference architecture. In the offline manager, the offline UI component generates the UI required during the offline mode, which is combined with data from the data processor to render to the user. The respondents were asked to opinion out the advantages and disadvantages of this approach to render the UI to the user to interact, as per question 8.

One of the respondents thought that there was no drawbacks for the component and indicated that offline UI's advantage is that it fetches the data from the cached data when there is a connection loss with the cloud. The respondent indicated that the offline UI component would render the UI to the user without any discrepancies during poor internet connection and cautioned that if there is database corruption or cached data corruption, then it might be difficult for the component to render the data.

The Senior developer felt that one of the drawbacks was how the rendering is achieved. He felt that while implementing the component, the requirement for additional memory and CPU might arise, which needs to be handled carefully. Another respondent was of the opinion that there might be an additional delay when there is a switch between online and offline mode. The Architect emphasized on memory management. The respondent indicated to look into the size of memory that will be utilized by the data processor and raised concern that, whether it will be sufficient enough to handle longer duration of offline state without affecting data loss.

The respondents were asked a follow-up question about their opinion of the offline renderer to render UI and how it might affect state synchronicity and latency, as per question 9.

The Senior developer was of the opinion that, though Offline renderer was useful in creating realistic images and videos from the processed data, it will have a huge impact on the latency of the system when compared to the online connection state between cloud and infotainment.

Few respondents thought that the component might not affect latency and felt it was a good idea to move the state out of the equation. But they also suggested that, in order to improve performance, it might be a good idea to consider merging online mode and offline mode rendering, so that the need to transfer the amount of data is reduced. They also stated that offline renderer was a good approach to render the data.

5.2.2.3 User Manager component

The respondents were informed about the additional responsibility added to the User Manager component, as this component already exists in AGL architecture. They were asked to give out their opinion on possible latency this component might add due to the additional responsibility of redirecting the request based on internet connection(online/offline mode) as per question 10.

Figure 5.14 illustrates that 60% of the respondents felt that latency added will be low and will be between 1-5ms, whereas 40% felt the latency added will be between 5-10ms. There was a follow-up question for this, which asked the motivation or reasoning for their responses.

Few respondents were of the opinion that the latency added will be less as this is an existing component in AGL architecture and it should not affect latency much, while another respondent answered that this component would lead to high latency as it will be overloaded with additional responsibility. One respondent felt it depended purely on implementation and thought it might be less than 5ms, but was unsure. The Software architect was of the opinion that this additional responsibility might add more latency. The respondent specified that this component would redirect the request based on the internet connection and indicated that this would lead to the addition of few more components, and it might increase the overall latency of the system.

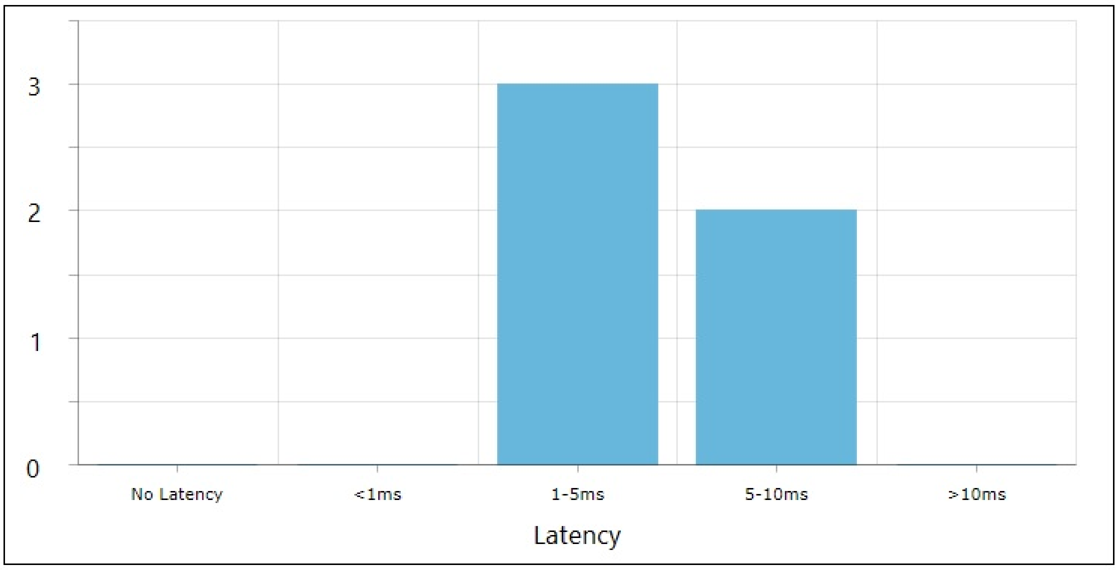


Figure 5.14: Latency added by User Manager

5.2.2.4 Other questions on architecture

Following the component-specific questions, the researchers asked generic questions on architecture. The respondents were asked if the components in reference architecture were overloaded with responsibilities, to which many respondents stated that they were overloaded with responsibilities. Follow up question was asked to identify which component was overloaded and, if we divide the component into two or more sub-components, will it have an impact on latency as per question 11.

Figure 5.15 depicts that for Offline Manager, all the respondents felt that dividing the component into two or more sub-component will decrease the latency of the component. For the Infotainment-Cloud State Sync Manager component, 60% felt that dividing the component will decrease the latency, while the remaining 40% were of the opinion of not dividing the component. In the case of the User Manager component, similar to Offline Manager, all the respondents felt that dividing the component into two or more sub-component will decrease the latency of the component. They felt that it was overloaded and might add high latency. Dividing it into more sub-components might reduce the impact on latency. For the Policy Manager component, 80% felt that dividing the component will decrease the latency, while the remaining 20% were of the opinion of not diving the component.

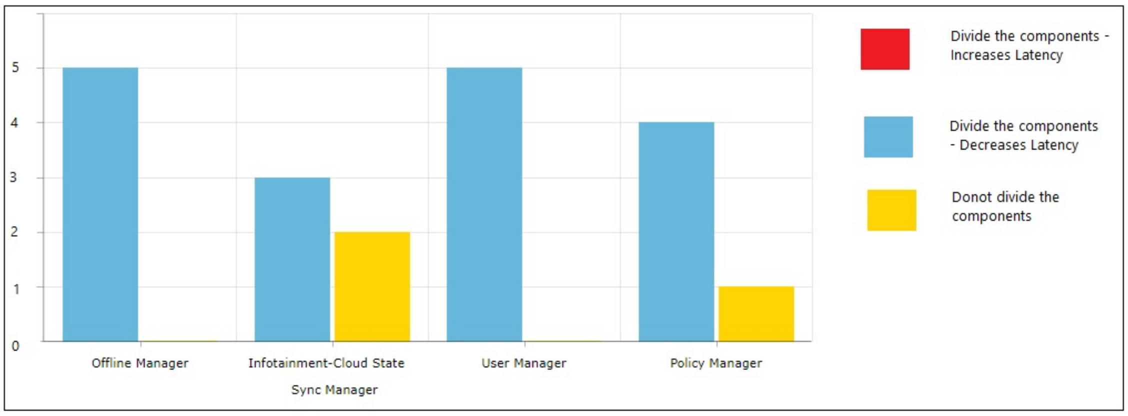


Figure 5.15: Impact of latency on dividing the component

The respondents were asked about the features that work well in the proposed reference architecture and the reason behind their opinion. Most of the respondents stated that the Infotainment-Cloud State Sync Manager component was a good idea, as it makes state management simpler. They felt it could make the user experience better. One of the respondents thought that the reference architecture is good enough to achieve real-time data streaming in infotainment system with low latency.

The respondents also stated that the implementation of the components would play a crucial role in determining the latency. They were of the opinion that with good implementation, this architecture might add good value for real-time data streaming in the automobile industry.

The researchers asked the respondents for suggestions to make the reference architecture more efficient in terms of Performance, Availability, and Security attributes. They emphasized on exploring more on the security of data, while acknowledging that User Manager encrypts the data used for communication with cloud and user authentication, which provides good data security. They highlighted that data from systems like User Profile, Advanced Driver Assistance Systems (ADAS) must be handled with utmost care. In terms of performance, one of the respondents indicated that having a simple implementation will be helpful to make a real estimation regarding latency. The respondent also pointed out that giving more details about each component's working will help in reasoning out the latency estimates.

The respondents indicated that though the estimates they have given on latency of the components are based on experience, knowledge and component description, they cautioned that they could change based on implementation. They also answered that the offline mode feature is a good idea for availability attribute during the loss of internet connection. They also asked us to ponder over how the failure of the Offline manager component will be handled and how it can affect the availability.

To conclude the interview, researchers asked for other overall suggestions. Few respondents replied that we must also look into how component failure can be handled and what is its impact on the system, while few did not have any more suggestions. One of the respondents was of the opinion that our contribution to the thesis is good. He was very happy with our thesis, and he thought it would be a good value addition to AGL.

5.2.3 Research Question 3

To answer the research question "How can the AGL reference architecture be extended to support near real-time, low-latency, high data volume content streaming?" results of the literature review of existing architecture are discussed.

A review of the existing AGL architecture specification determined that the current specification does not contain the roadmap features required to achieve near real-time low latency high data volume content streaming. However, the existing specification supports Connectivity and OTA. But, the cloud and content streaming applications are not present in any layer and are also not specified in the current roadmaps of the specification document.

The literature review results show that AGL is the most appropriate in-vehicle platform over other potential platforms because of platform features, platform run-time, application run-time, application development, SDK, app store, and licensing, and developer community. In order to extend the AGL architecture, multiple factors were identified and reviewed. Firstly, the researchers had to identify in which layer of the architecture does the features or elements like cloud, connectivity, OTA, or/and streaming-based functions need to be implemented. The researchers also had to identify whether AGL architecture can support the required technical feasibility required for real-time low latency high data volume content streaming.

Considering the factor of identifying the layer on which the elements need to be implemented, multiple literature and AGL meetings were reviewed. According to Zhang and AGL summer meeting [82] [83], AGL supports a number of Vehicle-to-Cloud (V2C) scenarios, which can be enabled by connecting AGL to the cloud. According to the AGL specification, Connectivity is in the service layer, and OTA in the Application layer [80]. According to [82], the AGL cloud is implemented in the application framework layer, where it provides methods to create Software applications and their user interface. Streaming is also one type of V2C scenario. The AGL application layer provides basic services to all applications regardless of the framework they are implemented so that there is a standard method providing the services. It makes logical sense to have cloud and streaming in the application layer so that they provide standards methods to access cloud and streaming features irrespective of frameworks.

Technical requirements of different applications were reviewed in order to find whether the streaming of real-time low latency contents can be supported in AGL. Initially, Apple Carplay and Android Auto's software platforms were reviewed because these software platforms mirror the smartphones using wifi and USB connection, which is used as a third-party application in the infotainment unit. To support Apple Carplay and Android Auto, the OEMs provide a minimum technical requirement for the infotainment system, which is compared with the technical requirement of AGL, to find whether applications used in Apple Carplay and Android Auto can be implemented in AGL.

To support standard applications like Spotify, Google stadia by AGL, the researchers reviewed the minimum technical requirements of Google Stadia to support streaming through USB from phone to Infotainment systems. It was found that Google Stadia requires an operating system of Android 6.0 or later and bandwidth requirement of 35Mbps for 4K with HDR and 5.1 surrounds, 20Mbps for 1080p and 5.1 surround, and 10Mbps for 720p and stereo sound. The limitation associated with playing Google Stadia on a mobile data network is that the user must be on an LTE or 5G network. The mobile device must have a solid, consistent cellular signal and should stay in one place. If you're in a vehicle or moving quickly, your connection may be interrupted. The user needs to avoid public areas with lots of mobile device network traffic. Network performance may be degraded in these areas.

Based on the above literature review, researchers were able to find the limitations in the current technologies and missing implementations in AGL. The limitations were the video streaming from the phone to the head unit requires HDMI support or the Chromecast (incase of google stadia). AGL does not provide a roadmap of technical requirements about how they might achieve high content low latency video streaming. The majority of OEMs are moving towards open source AGL rather than Genivi/QNX.

Once the limitations were identified, the researchers performed a literature review to find whether AGL provides support for high content, low latency video. Based on the review, it was found that the multimedia services are present in the Automotive services, which is a part of the Services layer in AGL architecture. AGL provides an API that allows the handling of various media data within the system, including audio/video playback, APIs to support QoS and recording, and media streaming over the network. AGL provides support for major network streaming protocols such as HTTP, RTSP, Digital Radio (DAB), DigitalTV, and it is also possible to extend the set of supported streaming protocols. AGL also provides support for major multimedia containers, such as MPEG2-TS/PS (ISO/IEC 13818-1), MP4 (MPEG-4 Part 14, ISO/IEC 14496-14:2003). It is also possible to extend the set of supported multimedia formats according to system requirements and extend AGL to support additional optional multimedia containers such as OGG (RFC 3533), 3GPP (ISO/IEC 14496-12). In addition to the above specification, AGL also supports Media Audio Codecs such as MP3 (MPEG-1/2 Audio Layer-3, ISO/IEC 11172-3, ISO/IEC 13818-3), AAC (ISO/IEC 13818-7, ISO/IEC 14496-3). It is also possible to extend AGL to support additional audio codecs, such as VORBIS [85], Windows Media Audio. AGL also supports Media Video Codecs such as MPEG-2 (ISO/IEC 13818-2), MPEG-4 Part 2 (ISO/IEC 14496-2), H.264 (MPEG-4 Part10, ISO/IEC 14496-10, ITU-T H.264). It is also possible to extend the set of supported audio and video codecs in accordance with system requirements. Therefore, the review shows that AGL supports the necessary technical specification required to stream real-time low latency high contents.

Once the researchers found that AGL contains the necessary technical specification to support near real-time high content low latency streaming. In the next step, the researchers reviewed the architecture, and the components which are used in the AGL architecture [80]. The current AGL architecture was reviewed to identify the components that can be reused, where the existing components are provided with additional responsibilities to support content streaming. On reviewing the existing AGL architecture, the researchers identified that policy manager and user manager could be modified by adding more responsibilities to support content streaming during online and offline modes. In addition to these components, additional components were identified and added to different layers of the AGL architecture in order for the infotainment to support the streaming during no or low internet connection.

In order to provide a seamless user experience, the infotainment system needs to stream the data to users irrespective of the internet connection. The new changes include new components like Offline manager, Infotainment-cloud state sync manager, data encryption. The identified components are placed in the different layers of the current AGL, as discussed previously in the section. The components Offline manager and Infotainment-cloud state sync manager are placed in the application framework layers of the AGL architecture. These components help the system in different ways, Firstly with the help of the offline manager, the system can stream the data to the user when there is a drop in connection. With the help of an offline manager, the system can cache the data from the cloud and can stream the data to the UI in case of loss of network connection. Secondly, with the help of the Infotainment-cloud state sync manager, the system can update the state of the UI data request from the cloud. Apart from its existing responsibility, the user manager provides the component to check the connection at regular intervals between the infotainment system and the cloud. Whereas the policy manager acts as a decision manager where it decides whether the requests need to be sent to the cloud for the response or whether it can be handled locally.

To conclude, the researchers were able to make changes to the existing AGL architecture, and with the help of the changes, the researchers were able to propose a new reference architecture that can be used to stream near real-time, high content, low latency content data from the cloud irrespective of the internet connection.

6 Discussion

This section presents the discussion on factors that helped in developing an architecture along with the validity threats. The first section presents the tactics used to achieve quality attributes for the architecture, the second section presents the discussion on how the functional requirements were achieved, the third section presents the reflection on architecture designed, the fourth section presents the reflection on methodology used, and the final section presents the threats to validity.

6.1 Quality attributes tactics

Architectural tactics are the techniques an architect uses to achieve quality attributes. They are the design decisions that influence the achievement of a quality attribute response. An architectural tactic is a means to satisfy the quality attribute response measure by manipulating the model through design decisions [61][65]
.
The system is designed based on the number of decisions. These decisions help in controlling the quality attribute responses, while some decisions ensure the system functionality is achieved. Tactics are like design patterns that intend to control responses to stimuli. Applying design patterns is often difficult due to the complexity of the design patterns as they consist of a bundle of design decisions. The quality attribute is achieved by modifying and adapting the design patterns, and applying the tactics helps in assessing the options to add details to the existing patterns. Tactics allow us to construct design fragments from "first principles" when no design pattern exists. Tactics make the design more systematic. The choice of which tactic to use depends on a trade-off among the quality attributes.

According to the data collected from the interview and brainstorming sessions, the quality attributes the architecture developed should achieve are availability, security, and performance. In the below section, we present the tactics of the three quality attributes used in developing the architecture.

6.1.1 Availability tactics

Availability is one of the quality attributes which refers to the ability of the system to be available at any point, irrespective of failures or outages [65]. One of the main challenges of this study is that the IVI system needs to be always available for the user even when there is a drop in connections. This challenge is achieved by using

Availability tactics in the developed architecture.

Consider the scenario; the infotainment system loses the network connection when there is a data request. We have used the Ping/echo tactic in the user manager component to detect the connection between the Infotainment system and the cloud system. On detecting connection drop or loss, the user manager keeps checking the network every second by sending the ping request to the cloud system [65].

We have used active redundancy in case of network drop. Consider the scenario, cars will be moving long distances, and we will not be aware of when there will be a connection drop or a poor connection. When there is a connection drop, the system must be able to provide the user with data all the time. The user manager component is responsible for detecting this fault and uses a ping tactic to monitor the network connection. On failing to connect with the cloud system, the user manager sends the request to the Offline manager. The Offline manager handles the request and sends the response to the UI. Hence in offline mode, the user can use the IVI system. The interview results indicated that having an Offline Manager component to provide data to UI during offline mode was a good idea. They believed that this would make the UI available to the user in offline mode.

In addition to this, we have also used a secondary database. The secondary database contains the same information as the primary but is maintained in a separate server. The secondary database is used when the primary database fails or crashes. The database is used to respond to data requests made by the Offline manager.

6.1.2 Security tactics

Security refers to the ability of the system to protect the data and information from any unauthorized access and enabling authentic users to use the system without any restriction [65]. One of the limitations of cloud technology is the lack of security. In this study, this limitation is mitigated using security tactics. The security tactics used are encrypt/decrypt data and authenticate users.

Consider the scenario, authentication of the cloud system for the users. Users have to be authenticated and will contain a single active session. This will be performed via an SSL certificate for increased security. Using the Authentication manager and SSL certificate, the risk of intrusion or spoofing is controlled as the SSL certificate is provided with a signature validated by the authentication manager in the cloud.

The response and the request information to and from the IVI are encrypted using the data encryption component. By using this tactic, the information communicated between the systems is secured, thus restricting the system from unauthorized access. The interview respondents also emphasized the importance of security of the data shared between the infotainment and the cloud. They appreciated the encryption and authentication mechanism in place and indicated the data must be handled securely, and other security options can be explored.

6.1.3 Performance tactics

Performance is related to the timing requirements of the system. It is the ability of the system to respond well in time for the events, data requests from different systems [65]s. The main purpose of this thesis is to provide continuous streaming of data from the cloud to the user irrespective of the connection. The IVI system involves many user actions; due to this, there will be many information requests coming from the system to the cloud. The response from the cloud system needs to be presented to the user without any interruption. The performance tactics used are resource management, and multiple copies of databases are maintained.

Consider the scenario, a data request is sent, and during the response, there is a drop in internet connection. The response to the user is delayed, and the data in the infotainment system is in a different state compared to data sent from the cloud system action. To overcome this, we have used different components like Infotainment-Cloud State Sync Manager and Offline Manager, which stream the data to the Infotainment UI. The Infotainment-Cloud State Sync Manager stores the latest state of the Infotainment UI at the time of disconnection. When there is a loss of internet connection, the cached data in offline mode will be used to process Infotainment requests. On reconnection, the latest state data of UI will be sent by Infotainment-Cloud State Sync Manager to the cloud via User Manager. The cloud resolves the state and sends the updated state. The Infotainment-Cloud State Sync Manager receives the updated state, updates the current state, and sends it to the Offline Manager to update the UI. By assigning the state management to separate components, the overloading of the component with responsibilities is reduced. Having a dedicated component for a specific role will help in better management of the component, and interview respondents were of the opinion that the Sync manager component would add negligible latency to the overall architecture. By using these resources, the latency is predicted to be decreased.

6.2 Functional Requirements tactics

The functional requirement plays a vital role in the design of the architecture. Based on interview results for RQ1, navigation and music(audio/video player) were identified as the basic functional features that users expect to be part of the infotainment system. But after thorough literature study, researchers identified two important functional requirements for the architecture, which were state management and offline management.

State management was thought to be an important functional requirement, as it is very commonly used in the gaming industry. As the system faces changing internet connections, there was a need for state management. If the states are not managed properly, it might lead to state mismatch between the infotainment system and the cloud. To address this issue, researchers decided to add Infotainment-Cloud State

Sync Manager component to the architecture, which handles the state synchronization between online and offline mode. At regular intervals, this component will check with the User Manager that if there is a connection established to the cloud. On reconnection, the latest state data of UI will be sent by Infotainment-Cloud State Sync Manager to the cloud. The cloud receives this state, updates the state, and sends back the updated state. The Infotainment-Cloud State Sync Manager component uses the cloud state to update the current state of the infotainment system. The respondents of the interview for RQ2 appreciated the idea of state management and thought it was a good approach to handle the states separately. They felt this component would play a crucial part in user experience.

Offline management was thought to be necessary to handle the request when there is no internet connection(offline mode). When there is no internet connection, there was a need for a mechanism to manage the UI requests, without which the users might face issues in accessing the infotainment UI. This might lead to a bad user experience. In order to address this scenario, the researchers added Offline Manager component to the architecture. In the offline manager, the offline UI component generates the UI required during the offline mode, which is combined with data from the data processor to render data to the user. The interview respondents of RQ2 were of the opinion that this component would render the UI to the user without any discrepancies during poor internet connection. They felt that the offline renderer was useful in creating realistic images and videos from the processed data but cautioned that there might be an additional latency added to the system due to the component.

6.3 Reflection on Related work

In the related work section, researchers presented currently trending cloud applications like Spotify and Google Stadia, open source platform Automotive Grade Linux (AGL), software platforms Apple CarPlay and Android Auto. All of these applications vary with our thesis on different factors , from user behavior, the size of streaming objects, and the number of objects offered for streaming. The researchers identified that in automotive industry, there are limitations in streaming high content, low latency video from cloud to infotainment.

Also, researchers identified that there were lot of papers on AGL and how it can be customised, but noticed that there was lack on papers on implementing cloud content streaming in AGL. Due to this, there is room for research regarding the different approaches that can be used to stream real-time, low-latency, high data volume content from cloud to infotainment. On contrary, studies regarding the architecture evaluation methods were common. The researchers discovered that evaluating the architecture based on semi structured interviews was best suited for getting architecture evaluated by multiple people who had required background knowledge. It provided the researchers to interact with the respondents and gather their insight about the proposed architecture.

6.4 Reflection on Architecture

The reference architecture was designed and developed based on the literature study and interview results from the first phase of data collection. The architecture was designed by keeping in mind the quality attributes that were prioritized by the stakeholders.

The architecture developed was evaluated using interviews. Based on the results from the interview, the researchers analyzed that the proposed architecture served as a good starting point towards the right direction in achieving cloud-based near real-time, low-latency, high data content streaming to car platforms. The architecture addressed an important area of state synchronicity management when the system switches between offline and online internet connection mode. The results indicated that the idea of state synchronicity management received a positive response. The interview respondents highlighted that state synchronicity management is the need of the hour when the system switches between varying internet connections. They indicated that the idea of carving out the state synchronicity management to a separate component was a good design decision. They mentioned that many cloud-based gaming applications currently have state synchronization management techniques implemented in them, and by including it in the reference architecture, design of the architecture was thought to be in-line with the current cloud applications. To make the system available in offline mode, the Offline Manager component was designed, and the interview respondents were of the opinion that it was an interesting approach to make UI available in offline mode. The offline UI, which can be used in UI rendering, was thought to be useful. They also indicated that identifying and using existing components of AGL is always a good approach, as it promotes reusability. Adding additional responsibility to the User Manager was thought to be useful, as the added responsibility was closely linked with its existing responsibility, and grouping them was a good design approach. It was thought to be the same with the Policy manager component as well, which is used to redirect the request as it is closely linked with its existing responsibility.

The positives of the architecture being stated, there were a couple of areas which architecture needed to address or improve on. The architecture did not furnish further details on how new states added to the system are being tracked and how the state switch is handled. The architecture also needed to consider the amount of resources like CPU, memory, that will be needed by the Offline Manager component during offline mode. One of the important feedback for architecture was regarding the latency. The interview respondents indicated that there might be an additional delay when there is a switch between online and offline mode. They suggested that to improve performance, it might be a good idea to merge online mode and offline mode rendering, so that the need to transfer the amount of data is reduced. They indicated that a couple of components might have a sizeable impact on latency. Components like Offline Manager and User Manager was thought to add more latency due to their role/responsibilities, whereas Infotainment-Cloud State Sync Manager component was thought to have less impact on latency. They indicated that it's always good to design components that have a negligible impact on the latency of the system. The overall end-to-end latency was thought to be less and within 10-50ms, which is good for the Infotainment system. The general opinion was that most of the components in architecture were overloaded with roles/responsibility, and it was suggested to divide them into two or more sub-components to reduce the latency caused by the components. Exploring different ways to redesign the components to reduce latency was suggested.

The interview feedback also indicated that latency results predicted for the architecture might change in real-time based on the type of implementation. The architecture needed to address the issue of component failure and its impact on the usability of the overall system.

Overall, the reference architecture identified and addressed the key areas needed for achieving cloud-based near real-time, low-latency, high data content streaming to car platforms, with enough room to improve it to handle the areas that it missed. The architecture was thought to provide a good infotainment experience with the right implementation.

6.5 Reflection on Methodology

The research methodology used for this thesis was Design Science Research Methodology (DSRM). The researchers were of the opinion that DSRM aided their thesis in the right direction. The iterative approach and specific guidelines helped researchers to plan and execute the thesis in a time-bound manner.

Interviews, Brainstorming, and document study was used for data collection. Document study helped researchers to gain good domain knowledge and identify the areas of improvement in the domain. Brainstorming helped them to think about possible issues that might require attention when developing the architecture. Interviews provided the researchers a good opportunity to interact with people who had the domain knowledge and work experience. It provided them with valuable information that helped researchers in developing the architecture. Interviews were conducted to evaluate the architecture developed with respondents who had prior knowledge of AGL. Though it provided good evaluation points, researchers felt they had to include another methodology for evaluation as well. They felt having more than one methodology for the evaluation would have improved their thesis results.

6.6 Threats to validity

The validity of the study illustrates the reliability of the results, the truthfulness of the results, and the unbiased subjective viewpoint of researchers[30]. The main validity threats that researchers predict for this study are Conclusion validity and External validity. In this section, the above-mentioned threats are discussed, followed by possible measures availed to reduce their impact.

6.6.1 Construct Validity

Construct validity is mainly focused on the early stages of the project. It is related to how the research is being conducted. "Construct validity concerns generalizing the result of the experiment to the concept or theory behind the experiment "[28].

In order to evaluate our architecture in a better way, researchers chose interviews for the evaluation process. The evaluation results derived from it might vary if researchers had chosen other evaluation methods. Moreover, the prototype tool developed was designed to focus on a small section of the online-offline scenario. The tool developed was not a very extensive one, ignoring many other areas. The tool was developed and tested on the same system. The change in network connection was simulated. These produce a risk to construct validity, as the results obtained might differ in real-world scenarios. The result obtained might be affected when the tool is put to use in a real-world scenario.

6.6.2 Internal Validity

Threats to internal validity are things that can affect the independent variable with respect to causality, without the researcher's knowledge[28].

For conducting the interviews, researchers chose a set of software engineers working in Aptiv, and for evaluation, researchers chose people who had prior knowledge or work experience in AGL. The data obtained from the interview were used for the result. It might be possible that if a chosen group of volunteers were more diverse (job/experience/work geography/gender), the result obtained might have been different. Including more volunteers might have affected the results. The selected volunteer group is not representative of the whole population. There was one volunteer(software engineer working at Aptiv) who was part of both interview and architecture evaluation. The volunteer, who was part of the interview, was already familiar with the type of architecture being developed. The volunteer's prior knowledge might have influenced his feedback in architecture evaluation. If different group of volunteers were chosen for the evaluation, the results might have changed.

6.6.3 Conclusion Validity

Conclusion validity relates to the process of analyzing data, finding patterns in the data, and drawing conclusions from it. Conclusion validity is more of our ability to draw correct conclusions from our observations.[29]

The volunteers were interviewed in a semi-structured format, where a couple of questions were open-ended. There might have been a possibility that volunteers would have misinterpreted the question and given incorrect responses. The thesis work is carried out in a company based on a non-disclosure agreement; due to this; it might be difficult to reproduce the same results outside because of missing and confidential information.

The reference architecture developed might not be the best one created, as the researchers had many trade-offs to handle and took the decision based on their knowledge, experience, and available data. Different options in trade-offs design decisions might have impacted results.

6.6.4 External Validity

Threats to external validity are conditions that limit our ability to generalize the results of our experiment to industrial practice.[28]

The set of software engineers chosen for interviews were from Aptiv and did not represent the whole population. Due to time constraints, only a limited number of volunteers were involved in data collection. If more people were included, the data obtained from interviews might have been different and might have impacted the result. For architecture evaluation, the volunteer group consisted of very few people who had prior knowledge about AGL. If we had included more people working on AGL projects, then the evaluation result might have differed. The researchers chose to build the reference architecture based on AGL architecture, as it was more popular and widely used.

If researchers had chosen other infotainment platforms used by automakers like QNX, GENIVI, Windows Embedded Automotive, Android Automotive, then the reference architecture built might have been different, and results would have differed. The evaluation method used was Interviews. If other evaluation methods like ATAM were used, the results obtained might have been different. These factors increase the threats to generalizing the results.

7
Conclusion

This thesis was conducted with the aim of understanding the issues involved in enabling cloud-based near real-time, low-latency, high data content streaming to car platforms that provides high-end user experience, and design a reference architecture that could achieve the real-time, low-latency, high data content streaming.

The study is closely related to the research questions (RQ1, RQ 2, and RQ 3), and the answers to the respective questions can be found below section.

7.1 Answer to Research questions

RQ 1:

What are functional and non-functional attributes that are required for the reference architecture to support near real-time, low-latency, high data volume content streaming?

Based on data analysis results from Figure 5.7 and literature study, the characteristics and features, which researchers have identified are : Function features - State management, Offline management and Non-functional features - Performance, Availability and Security. The functional features proposed in the architecture was appreciated by the interview respondents and they indicated that it will play crucial role in achieving good user experience. These attributes (functional and non-functional) are most necessary for developing reference architecture, which will serve as a guide to building a next-generation infotainment system, streaming the complete infotainment system from cloud to in-car.

RQ 2:

How can we evaluate that the attributes identified in RQ1 have been satisfied in the reference architecture developed?

The researchers have used the semi-structured interview to evaluate the attributes identified in the previous question. Using interviews, researchers successfully evaluated whether the reference architecture developed would satisfy the attributes deemed necessary for the architecture for near real-time, high content, low latency content streaming. The interview helped the researchers to evaluate each of the components in the infotainment system and identify how each component in the developed reference architecture influenced the attributes identified in RQ1. The interview helped the researchers to gather good insight about the proposed architecture from the respondents, who had prior knowledge of AGL. Based on the interview feedback, the reference architecture identified and addressed the important areas needed for achieving cloud-based near real-time, low-latency, high data content streaming to car platforms, but it also fell short in addressing few other areas. The architecture was thought to provide a good infotainment experience with the right implementation and had enough scope for improvement.

RQ 3:

How can the AGL reference architecture be extended to support near real-time, low-latency, high data volume content streaming?

The researchers have used a document study literature review to gather the data required to perform the current AGL architecture changes. On a thorough review of the literature, researchers identified that the current AGL architecture does not have any roadmap regarding content streaming. Using the literature review, researchers were able to successfully identify the technical feasibility required for the streaming near real-time , high content, low latency content streaming from the cloud. The researchers were able to identify the components which can be reused from existing AGL architecture and incorporate additional new components into the new reference architecture for content streaming, which is based on AGL architecture.

7.2 Future Work

This thesis can be considered as an initial step in understanding the issues involved in enabling cloud-based near real-time, low-latency, high data content streaming to car platforms, which provides high-end user experience and identifying the functional and non-functional attributes required to build a reference architecture for next-generation infotainment system. There are many other ways in which the research can be conducted in this thesis.

As indicated in the report, this thesis mainly focused on infotainment system architecture, its attributes, and evaluation. To identify and investigate more bottlenecks, different architecture approaches/styles for cloud part and infotainment part could be explored. The reference architecture was built on top of AGL architecture, but various other platforms like Windows Embedded Automotive, Android Automotive can be considered to build the reference architecture. Numerous architectural tactics could be inspected in achieving the attributes for the reference architecture. Different combinations and variations of offline/cache mechanisms could be explored for achieving interactive UI in offline mode. Numerous state synchronization management techniques can be explored. To further increase the generalizability of the result, other modes of transport(bus, truck, ship, etc.) can be pursued. To enhance the user gaming experience, ways of streaming cloud gaming applications like Google Stadia can be looked into. Technology like Holoride, which gives hyper-immersive experiences, comes pre-installed at present, but if it can be streamed from cloud to car, the XR experience provided will be better and up to date. Several ways to achieve this can be investigated.

7.2.1 Pathway to Implementation

The reference architecture developed can be used for the development of some exciting applications. Consider the Holoride application being used in the infotainment system, which uses the reference architecture developed. Holoride is an application that provides a dynamic VR experience for the passengers in cars. It provides passengers with fascinating theme games combined with vehicle data to move through a colorful fantasy of their own.

Consider the scenario where the passenger is bored during the travel and grabs the virtual reality headset. When there is a regular connection(online mode), the vehicle data is sent to the cloud and the game theme from the cloud is streamed to the user (virtual reality headset) through the policy manager. When the user experiences a drop in network connection, the user manager detects and sends the request to the offline manager. A lightweight version of the game can be stored in offline manager or local database, which can be used during offline mode. The offline manager checks for the cache data or local database and renders the data response to the user who can continue the game without any interruptions. The Infotainment-cloud state sync manager keeps track of the latest state and checks for the connection regularly through the user manager. On reconnection, with the help of the Infotainment-cloud state sync manager, which keeps track of the latest state of the game, the online game's state is resolved, and the updated state of the game is rendered to the user via the offline manager. In this way, we can provide a seamless user experience.

Bibliography

[1] M. Galster, "Software reference architectures: related architectural concepts and challenges," 2015 1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures (CobRA) , Montreal, QC, 2015, pp. 1-4

[2] Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice. Addison Wesley, 3rd edition, 2013.

[3] R. Shea, J. Liu, E. C. -. Ngai and Y. Cui, "Cloud gaming: architecture and performance," in IEEE Network, vol. 27, no. 4, pp. 16-21, July-August 2013, doi: 10.1109/MNET.2013.6574660.

[4] S. Chuah, C. Yuen and N. Cheung, "Cloud gaming: a green solution to massive multiplayer online games," in IEEE Wireless Communications , vol. 21, no. 4, pp. 78-87, August 2014.

[5] Everton Cavalcante, Marcelo Pitanga Alves, Thais Batista, Flavia Coimbra Delicato, and Paulo F. Pires. 2015. An Analysis of Reference Architectures for the Internet of Things. In Proceedings of the 1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures (CobRA '15).

[6] Samuil Angelov, Jos J.M. Trienekens, and Paul Grefen. Towards a method for the evaluation of reference architectures: Experiences from a case. In Second European Conference, ECSA, Software Architecture, pages 225–240. Paphos, Cyprus, September 29-October 1 2008.

[7] Hyunwoo Nam, Kyung-Hwa Kim, Doru Calin and Henning Schulzrinne. YouSlow: a performance analysis tool for adaptive bitrate video streaming. SIGCOMM '14: Proceedings of the 2014 ACM conference on SIGCOMMAugust 2014 Pages 111–112https://doi.org/10.1145/2619239.2631433

[8] Anthony Neal Park, Yung-Hsiao Lai and David Randall Ronca. Encoding video streams for adaptive video streaming. US20110268178A1, 2011-11-03.

[9] [Online] Available: Google Stadia `https://en.wikipedia.org/wiki/Google_Stadia` , Date of access: 2020-2-20.

[10] [Online] Available: A First Look at Google Stadia `https://dzone.com/articles/a-first-look-at-google-stadia` , Date of access: 2020-2-20.

[11] [Online] Advancement in infotainment system in automotive sector with vehicular cloud network and current state of art `https://search.proquest.com/docview/2331661420?pq-origsite=gscholarfromopenview=true` ,Date of access: 2020-2-17.

[12] Peter van der Stok. Dynamic and Robust Streaming in and between Connected Consumer-Electronic Devices. Springer, 2005

[13] G. Macario, M. Torchiano and M. Violante, "An in-vehicle infotainment software architecture based on google android," 2009 IEEE International Symposium on Industrial Embedded Systems, Lausanne, 2009, pp. 257-260, doi: 10.1109/SIES.2009.5196223.

[14] M. Klecha and S. Drude, "System Architecture for a modular and distributed Solution for next Generation Car Infotainment Systems," 2007 Digest of Technical Papers International Conference on Consumer Electronics, Las Vegas, NV, 2007, pp. 1-2, doi: 10.1109/ICCE.2007.341454.

[15] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole and M. Bone, "The Concept of Reference Architectures", Systems Engineering, vol. 13, no. 1, pp. 14-27, 2010

[16] M. Guessi, L. B. R. Oliveira, L. Garcés and F. Oquendo, "Towards a formal description of reference architectures for embedded systems," 2015 1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures (CobRA), Montreal, QC, 2015, pp. 1-4.

[17] G. Kreitz and F. Niemela, "Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming," 2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P), Delft, 2010, pp. 1-10, doi: 10.1109/P2P.2010.5569963.

[18] Roel J. Wieringa. Design science methodology: For information systems and software engineering. Mar. 2014, pp. 1–332. isbn: 9783662438398. doi: 10.1007/978-3-662-43839-8. url: https://link.springer.com/book/10.1007/978-3-662-43839-8.

[19] Guido L. Geerts, A design science research methodology and its application to accounting information systems research, International Journal of Accounting Information Systems, Volume 12, Issue 2, 2011, Pages 142-151, ISSN 1467-0895, https://doi.org/10.1016/j.accinf.2011.02.004. (http://www.sciencedirect.com/science/article/pii/S1467089511000200)

[20] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger Samir Chatterjee (2007) A Design Science Research Methodology for Information Systems Research, Journal of Management Information Systems, 24:3, 45-77, DOI: 10.2753/MIS0742-1222240302

[21] Jay F. Nunamaker Jr., Minder Chen Titus D.M. Purdin (1990) Systems Development in Information Systems Research, Journal of Management Information Systems, 7:3, 89-106, DOI: 10.1080/07421222.1990.11517898

[22] Alan R. Hevner, Salvatore T. March, Jinsoo Park and Sudha Ram MIS Quarterly.Management Information Systems Research Center, University of Minnesota. Design Science in Information Systems Research. Vol. 28, No. 1 (Mar., 2004), pp. 75-105 DOI: 10.2307/25148625 https://www.jstor.org/stable/25148625 Page Count: 31

[23] Paweł Weichbroth.Facing the brainstorming theory. A case of requirements elicitation, Journal of Studia Ekonomiczne. Publisher -Wydawnictwo Uniwersytetu Ekonomicznego w Katowicach. 2016 |Vol 296 | 151-162.

[24] U. Rafiq, S. S. Bajwa, X. Wang and I. Lunesu, "Requirements Elicitation Techniques Applied in Software Startups," 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, 2017, pp. 141-144, doi: 10.1109/SEAA.2017.73.

[25] "Prototype." UXL Encyclopedia of Science, edited by Amy Hackney Blackwell and Elizabeth Manar, 3rd ed., UXL, 2015. Gale In Context: Science

[26] [Online] Available: https://github.com/llfjfz/LiveRender

[27] [Online] Available: https://en.wikipedia.org/wiki/Qt_Creator

[28] Claes Wohlin et al. Experimentation in Software Engineering. Vol. 9783642290. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–236. isbn: 978-3-642-29043-5. doi:10. 1007/978-3-642-29044-2. url: http://link.springer.com/10.1007/978-3-642-29044-2.

[29] Miroslaw Staron. Action Research in Software Engineering. Cham: Springer International Publishing, 2020. isbn: 978-3-030-32609-8. doi: 10.1007/978-3-030-32610-4. url:http://link.springer.com/10.1007/978-3-030- 32610-4.

[30] Runeson, P., Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering, 14(2), 131-164.

[31] [Online] Available: Android Auto https://www.android.com/auto/

[32] [Online] Available: Apple Carplay https://www.apple.com/ios/carplay/

[33] Development of a Mobile News Reader Application Compatible with In-Vehicle Infotainment. Cham: Springer International Publishing, 2020. isbn: 978-3-319-97162-9. doi: 10.1007/978-3-319-97163-6_2. url: https://link.springer.com/chapter/10.1007/978-3-319-97163-6_2.

[34] Platforms and Ecosystems for Connected Car Services. Micha Bosler, Christopher Jud,Georg Herzwurm. https://tutcris.tut.fi/portal/files/13887121/Proc_IWSECO_2017.pdfpage=24

[35] [Online] Available: https://developer.apple.com/videos/play/wwdc2016/723/

[36] [Online] Available: https://developer.apple.com/videos/play/wwdc2017/717/

[37] [Online] Available: https://support.google.com/stadia/answer/9338852

[38] [Online] Available: https://support.google.com/stadia/answer/9578631

[39] [Online] Available: Google Stadia https://stadia.google.com/

[40] [Online] Available: https://www.android.com/auto/compatibility/#compatibility-vehicles

[41] [Online] Available: https://www.reddit.com/r/Stadia/comments/fnnnbm/whyhow_does_stadia_work_great_while_casting_my/

[42] [Online] Available: https://www.reddit.com/r/Stadia/comments/glxi85/will_casting_the_stadia_app_from_your_phone_to/

[43] [Online] Available: Chromecast https://store.google.com/product/chromecast

[44] [Online] Available: AGL Roadmap https://wiki.automotivelinux.org/agl-roadmap

[45] [Online] Available: https://www.orioninc.com/articles/agl-session-replay-from-all-member-summer-2020-meeting/

[46] [Online] Available: Spotify streaming https://support.spotify.com/us/article/high-quality-streaming/

[47] [Online] Available: Spotify system requirements https://support.spotify.com/us/article/spotify-system-requirements/

[48] Online] Available: Spotify https://support.spotify.com/us/article/storage-and-data-information/

[49] [Online] Available: Holoride https://www.holoride.com/

[50] [Online] Available: XR Eeality https://en.wikipedia.org/wiki/X_Reality_(XR)

[51] Klauer SG, Guo F, Sudweeks JD, Dingus TA. An Analysis of Driver Inattention Using a Case-crossover Approach on 100-car Data. Washington, D.C.: National Highway Traffic Safety Administration; 2010. Report No.: DOT HS 811 334

[52] Van Gog, T., Paas, F., Savenye, W., Robinson, R., Niemczyk, M., Atkinson, R., ... Hancock, P. A. (2008). Data collection and Analysis. Handbook of Research on Educational Communications and Technology 3e, 763-806.

[53] Galesic, M., Bosnjak, M. (2009). Effects of questionnaire length on participation and indicators of response quality in a web survey. Public Opinion Quarterly, 73(2), 349-360.

[54] Cape, P. (2010). Questionnaire length, fatigue effects and response quality revisited. Survey Sampling International.

[55] M. A. Babar, L. Zhu and R. Jeffery, "A framework for classifying and comparing software architecture evaluation methods," 2004 Australian Software Engineering Conference. Proceedings., Melbourne, Victoria, Australia, 2004, pp. 309-318, doi: 10.1109/ASWEC.2004.1290484.

[56] M. A. Babar and I. Gorton, "Comparison of scenario-based software architecture evaluation methods," 11th Asia-Pacific Software Engineering Conference, Busan, South Korea, 2004, pp. 600-607, doi: 10.1109/APSEC.2004.38.

[57] Mattsson, M., Grahn, H., and Mårtensson, F.: 'Software Architecture Evaluation Methods for Performance, Maintainability, Testability, and Portability', QoSA 2006

[58] J. Bosch and P. Molin, "Software architecture design: evaluation and transformation," Proceedings ECBS'99. IEEE Conference and Workshop on Engineering of Computer-Based Systems, Nashville, TN, USA, 1999, pp. 4-10, doi: 10.1109/ECBS.1999.755855.

[59] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson and J. Carriere, "The architecture tradeoff analysis method," Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193), Monterey, CA, USA, 1998, pp. 68-78, doi: 10.1109/ICECCS.1998.706657.

[60] B. Gallagher, "Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study," SEI, Carnegie Mellon University, CMU/SEI-2000- TN-007, 2000

[61] F. Bachmann, L. Bass, and M. Klein, "Deriving Architectural Tactics: A Step Toward Methodical Architectural Design," Software Engineering Institute CMU/SEI-2003-TR-004, 2003

[62] C. Wulf, C. C. Wiechmann and W. Hasselbring, "Increasing the Throughput of Pipe-and-Filter Architectures by Integrating the Task Farm Parallelization Pattern," 2016 19th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE), Venice, 2016, pp. 13-22, doi: 10.1109/CBSE.2016.21.

[63] Jones, L., Lattanze, A.: Using the architecture tradeoff analysis method to evaluate a wargame simulation system: A case study. Tech. rep., Software Engineering Institute, Carnegie Mellon University (2001)

[64] M. Barbacci, P. Clements, A. Lattanze, L. Northrop, and W. Wood. Using the Architecture Tradeoff Analysis Method (ATAM) to evaluate the software architecture for a product line of avionics systems: A case study. CMU SEI Technical Note CMU/SEI-2003-TN-012, Software Engineering Institue, Pittsburgh, PA, 2003

[65] Software architecture in practice I Len Bass, Paul Clements, Rick Kazman. 3rd ed. p. em. (SEI series in software engineering)

[66] I. M. Putrama, K. T. Dermawan, G. R. Dantes and K. Y. E. Aryanto, "Architectural evaluation of data center system using architecture tradeoff analysis method (ATAM): A case study," 2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA), Denpasar, 2017, pp. 1-6, doi: 10.1109/ICAICTA.2017.8090982.

[67] M. Saini, K. M. Alam, H. Guo, A. Alelaiwi, and A. El Saddik, "InCloud: A cloud-based middleware for vehicular infotainment systems," Multimedia Tools Appl., pp. 1–29, Jan. 2016, doi: 10.1007/s11042-015-3158-4

[68] S. Aust, "Paving the Way for Connected Cars with Adaptive AUTOSAR and AGL," 2018 IEEE 43rd Conference on Local Computer Networks Workshops (LCN Workshops), Chicago, IL, USA, 2018, pp. 53-58, doi: 10.1109/LCNW.2018.8628558.

[69] P. Sivakumar, R. S. Sandhya Devi, A. Neeraja Lakshmi, B. VinothKumar and B. Vinod, "Automotive Grade Linux Software Architecture for Automotive Infotainment System," 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2020, pp. 391-395, doi: 10.1109/ICICT48043.2020.9112556.

[70] Höttger R. Why Open Source is Driving the Future Connected Vehicle. Mobility in a Globalised World 2018. 2019 Jul 22;22:274.

[71] A. Castiglione, F. Palmieri, F. Colace, M. Lombardi and D. Santaniello, "Lightweight Ciphers in Automotive Networks: A Preliminary Approach," 2019 4th International Conference on System Reliability and Safety (ICSRS), Rome, Italy, 2019, pp. 142-147, doi: 10.1109/ICSRS48664.2019.8987693.

[72] G. Muller, A reference architecture primer, Gaudi project, 2008.

[73] [Online] Available: Automotive Grade Linux, `https://www.automotivelinux.org/software`, Date of access: 2020-11-16.

[74] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update", in Information and Software Technology, 2015, isbn: 0360-1315. doi: 10.1016/j.infsof.2015.03.007.

[75] [Online] Available: ACM Digital Library, 2020. `http://dl.acm.org/`.

[76] [Online] Available: Springer Link, 2020. `https://link.springer.com/`.

[77] [Online] Available: arXiv Link, 2020. `https://arxiv.org/`.

[78] [Online] Available: IEEE Explore, 2020. `http://ieeexplore.ieee.org/Xplore/home.jsp`

[79] [Online] Available: Cloud Gaming, 2020. https://786games.com/best-cloud-gaming-services-available-in-2020/

[80] [Online] Available: Automotive Grade Linux Requirements Definition. Automotive Grade Linux (AGL). 2020. https://docs.automotivelinux.org/en/master/#2_Architecture_Guides/1_Introduction/1_AGL_Requirements_Specifications/

[81] M. Ozcelikors and A. Gumuskavak, "Platform-independent Infotainment and Digital Cluster Development using Yocto Project," 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Istanbul, Turkey, 2020, pp. 1-5, doi: 10.1109/ICECCE49384.2020.9179288.

[82] [Online] Available: Orion Summer Meet 2020 https://www.orioninc.com/articles/agl-session-replay-from-all-member-summer-2020-meeting/

[83] Zhang, Q., Cheng, L. Boutaba, R. Cloud computing: state-of-the-art and research challenges. J Internet Serv Appl 1, 7–18 (2010). https://doi.org/10.1007/s13174-010-0007-6

[84] [Online] Available: Github for literature papers https://github.com/ramven-chalmers/Literature_review_papers

[85] [Online] Available: Vorbis http://xiph.org/vorbis/

A
Appendix 1

A.1 Semi structured Interview Questions for RQ1

Sl No	Interview Questions
1	What is your role in the company?
	a. How many years have you worked? How many years in infotainment domain?
2	Have you worked on the current infotainment system?
	a. If yes, What are the 3 key challenges you faced when working on current infotainment system?
	b. Prioritize the challenges with most important one as first
	c. What are the top quality attributes which you have implemented in the current infotainment system?
	d. How were the quality attributes of current infotainment system evaluated?
	e. What trade-offs were considered when implementing the quality attribute?
	f. What are the must have functional attributes when implementing a new module/component in the infotainment system?
	g. What are the must have non functional attributes when implementing a new module/component in the infotainment system?
	h. What were the key challenges faced when implementing the functional attributes?
	i. How were the functional attributes of current infotainment system evaluated?
3	What are the most important apps that you want in an infotainment systems?
4	Is it important that the infotainment system supports online/real-time gaming?
	a. If yes, how is it important?
5	Do you know about cloud based architecture?
	a. If yes, What are the 3 unique features in cloud architecture that stands out from other architectures?
	b. If yes,Can you name 3 key challenges you forsee when adopting a cloud architecture for infotainment system?
	c. Prioritize the challenges with most important one as first
	d. If yes, What are the 3 quality attributes you consider when adopting a cloud architecture for infotainment system?
	e. Prioritize the attributes with most important one as first
	f. What are the must have functional attributes when implementing a new module/component in cloud based system?
	g. What were the key challenges faced when implementing the functional attributes?
	h. How were the functional attributes evaluated?
6	Do you know about cloud gaming?
	a. If yes, What are the 3 unique features in cloud gaming?
II7	What are the apps which you use frequently while driving?
8	Other General Suggestions regarding the architecture we plan to develop?

A.2 Semi structured Interview Questions used for architecture evaluation

1. What is your role/responsibility? -

Junior Developer/QA
Senior developer/QA
Software Architects
Manager
Student
Other

2. Rate your knowledge/understanding of AGL

have just heard about AGL
have read whitepapers about AGL
have use AGL in prototype projects
have been involved in a product development based on AGL
have not heard about AGL

3. In the architecture design, we have provided a description about roles/responsibility of the new components and existing components in AGL. Rate your level of understanding of the components roles/responsibility. (Choose only one option per row)

Component	Level of Understanding
Offline Manager	
Infotainment-Cloud State Sync manager	
User Manager(with added responsibility)	
Policy Manager(with added responsibility)	

0- I do not understand the design
1- I understand the design , but would not feel comfortable to explain it to others
2- I understand the design and would feel comfortable to explain it to others
3- I understand the design and would feel comfortable to explain it to others, but could not make changes to adopt it for our needs
4- I understand the design and would feel comfortable to explain it to others, and could make changes to adopt it for our needs

4. According to the literature papers, the average end-to-end latency of interaction in an HMI system is around 100ms. According to you, rate the possible end-to-end data latency for UI interaction in our infotainment system?

Latency might be >150ms
Latency might be in the range of 100-150ms
Latency might be in the range of 50-100ms
Latency might be in the range of 10-50ms
Latency might be <10ms

5. In the proposed architecture design, the offline manager is responsible to render UI when there is no internet connection. When there is internet connection, the data is streamed from the cloud. Infotainment-cloud state sync manager component is used to synchronize the state between the cloud and infotainment system. What is your opinion on the state sync manager's role in achieving state synchronicity?(descriptive)

6. In the architecture, we have used the Infotainment-Cloud State Sync Manager component which is used to sync the state between infotainment and cloud. In your opinion, how much latency might this component add? Please provide the motivation for your choice as well.

Very High latency(>10ms)
High latency(5-10ms)
Low latency (1-5ms)
Negligible latency(<1ms)
No latency

7. In the architecture, we have used the Infotainment-Cloud State Sync Manager component which is used to sync the state between infotainment and cloud. In your opinion, do you think there is any drawback of this component in achieving state synchronicity? If so, what are the drawbacks and how it can be improved to achieve better state sync?(descriptive)

8. In the offline manager, the offline UI component generates the UI required during offline mode which is combined with data from the data processor to render to the user. In your opinion, What are advantages and drawbacks of this approach to render the UI to the user to interact? (descriptive)

9. In the offline manager, the offline UI component generates the UI required during offline mode which is combined with data from the data processor to render

to the user. What is your opinion about having an offline renderer to render UI? How do you think the offline renderer might affect state synchronicity and latency? (descriptive)

10.In architecture, we have used the existing User Manager component in AGL and added additional responsibility of redirecting the request based on internet connection(online/offline mode). In your opinion, how much latency might this add? Please provide the motivation for your choice as well.

Very High latency(>10ms)
High latency(5-10ms)
Low latency (1-5ms)
Negligible latency(<1ms)
No latency

11. In the architecture design, we have provided a description about roles/responsibility of the new components and existing components in AGL. If the components are overloaded with a lot of responsibilities and splitting the components might/might not have an impact on the latency, in your opinion, rate the impact of latency on component split.

Component	Split-Increases latency	Split-Decreases Latency	No Split
Offline Manager			
Infotainment-Cloud State Sync manager			
User Manager(with added responsibility)			
Policy Manager(with added responsibility)			

12. In your opinion, With our changes to the current AGL architecture, what features work well in the architecture and why? (Descriptive)

13. Do you have any suggestions to make the architecture more efficient in terms of Performance, Availability and Security?(Descriptive)

14. Any other suggestions?(Descriptive)